UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉNICA SUPERIOR

GRADO EN INGENIERÍA INFORMÁTICA

DETECTOR PATRONES DE COMPORTAMIENTO DEL CONDUCTOR

TRABAJO FIN DE GRADO

Autor: ÁLVARO MAROTO GÜENDIÁN

Tutor: AGAPITO LEDEZMA ESPINO

"He fracasado una y otra vez en la vida, por eso he conseguido el éxito"

Michael Jordan

Agradecimientos

Cinco años después de haberme embarcado en este viaje, al fin he aterrizado. Para no fracasar en el intento he tenido que apoyarme en toda esa gente que ha estado a mi lado desde el principio hasta el final, gente que siempre estará ahí.

Gracias a mis padres, Tito y Charo, por haberme educado desde pequeño y por haberme dado ese sentido de la responsabilidad desde que era bien pequeño. Gracias por costearme todos los estudios, por estar ahí cada minuto de mi vida y por ayudarme en todo lo posible. Nunca podré agradeceros todo lo que dais vosotros por mí.

Destacar por supuesto a mi hermana Sandra por cuidarme desde pequeño, por haber sido un espejo en el que mirarme y por apoyarme siempre que lo he necesitado. No me puedo olvidar de mi cuñado David, una de las mejores personas que he conocido en la vida y que tengo la suerte de compartir familia con él. Así mismo, gracias a los dos por haber dado vida a dos pequeñas preciosidades, como son mis sobrinos Dani y Alejandro, la alegría de la familia.

Este trabajo no habría sido posible sin mi tutor Agapito. A ti agradecerte haberme enseñado todos los entresijos de la Inteligencia Artificial. Gracias por haber sido tan buen tutor y profesor, ayudándome y guiándome en este proyecto. También mencionar a Óscar, compañero de departamento que tantas dudas me ha solucionado.

Llegar hasta aquí ha sido complejo, pero sé que habría sido imposible sin mis compañeros y amigos informáticos, especialmente Georgi, Carlos, Andrea, Rubén y Jorge. Gracias por ayudarme siempre que os he necesitado. Empezamos siendo compañeros y nos vamos siendo amigos para siempre.

Por último, y no menos importante, a mis amigos de sangre. A Javi, César, Guille, Richi y Pep. Curtidos en mil batallas y esperando a ver que nos deparará el futuro, pero siempre juntos. Gracias por compartir vuestra vida con la mía, por hacer que me pase el día riéndome y por apoyarme siempre que lo he necesitado.

Resumen

Los sistemas de ayuda a la conducción son cada vez más populares en el área de la automoción. Esto se debe a que mejoran notablemente la seguridad de los integrantes del vehículo, permitiendo, entre otras cosas, el frenado automático en caso de choque, el reconocimiento de señales de tráfico, el limpiaparabrisas automático en caso de lluvia, etc.

La mayoría de accidentes están provocados por distracciones del conductor, por lo que es necesario un sistema que permita detectar el comportamiento de los conductores. Para ello, se incorporan numerosos sensores en el vehículo, incluyendo cámaras que permiten obtener información tanto del interior como del exterior del vehículo.

Este Trabajo Fin de Grado se basa en el desarrollo teórico de un modelo de actividades, realizando asimismo una primera aproximación práctica, con el objetivo de sentar las bases de un modelo práctico que sea capaz de reconocer patrones de comportamiento de un conductor en tiempo real. Todo este estudio se apoya en un conjunto de cámaras y sensores simulados, los cuáles se encuentran incorporados en un simulador de coche. Mediante el uso de éstos se recogen todos los datos necesarios para llevar a cabo el presente Trabajo Final de Grado.

Palabras clave: Reconocimiento de Actividades, Visión por Computador, fichero log, trie, modelo de actividades, conjuntos de datos, STISIM, sensores.

Abstract

Advanced driver assistance systems are becoming increasingly popular in the automotive area. This is due to improve the safety of the members of a vehicle, allowing, among other things, automatic braking in case of crash, signal recognition, automatic wipers, etc.

Most of accidents are caused by driver distraction, so it´s necessary a system that is capable to detect the driver´s behavior. For this, a lot of sensors are incorporated in the vehicle, including cameras that enable get information both inside and outside the vehicle.

This final project is based on the theoretical development of an activity model. Also it is based in a first practical approach, with the goal to make a practical model, in the future, that will be capable of recognizing real time driver behavior patterns. This study is supported on a set of cameras and simulated sensors, which are incorporated in a car simulator. All of the data used in this project is collected by them.

Key Terms: Activity Recognition, Computer Vision, log file, trie, activity model, data set, STISIM, sensors.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1. Introducción

La Visión por Computador (*Computer Vision*) es una de las disciplinas de la Inteligencia Artificial que más ha crecido en los últimos años, basada en el estudio de como procesar, analizar e interpretar imágenes con el objetivo de obtener una determinada información que permita resolver problemas del mundo real [1] .

Uno de los desafíos de la Visión por Computador es el reconocimiento y clasificación de acciones o actividades humanas a partir de los fotogramas que componen una secuencia de vídeo, utilizando para ello técnicas de reconocimiento de patrones. El reconocimiento de actividades humanas por visión se compone, fundamentalmente, de tres tareas claramente diferenciadas: el pre-procesamiento del video, la representación de la información visual y el aprendizaje automático, combinado con técnicas estadísticas, para la detección patrones de comportamiento [2].

Una de las principales áreas de aplicación de la Visión por Computador son los sistemas avanzados de asistencia a la conducción, en los que el objetivo es reconocer la actividad o acción que está realizando en ese momento el conductor a partir de cámaras y sensores. Un sistema capaz de reconocer que actividad está realizando el conductor es de gran utilidad ya que puede evitar una gran cantidad de accidentes provocados por distracciones. Otras áreas de aplicación son la medicina, seguridad o industria.

1.1. Motivación

Los accidentes de tráfico son un auténtico drama para la sociedad actual. En los últimos diez años, se ha disminuido el número de accidentes en un 65% (*ver* Figura 1). Sin embargo, el número de accidentes y de fallecidos sigue siendo bastante alto. Durante el año 2014, se produjeron en España un total de 1131 fallecidos en 981 accidentes en vía interurbana [3].

Figura 1: Evolución accidentes de tráfico [4]

Numerosos estudios han demostrado que, aproximadamente, el 78% de los accidentes son provocados por distracciones del conductor, las cuales pueden ser físicas (distracciones con otros pasajeros u otros coches) o cognitivas (distracciones por estrés o tarea demasiado cotidiana y/o aburrida).

Un sistema capaz de reconocer si un conductor se encuentra distraído (por ejemplo, mirando la radio) evitaría una gran cantidad de accidentes y salvaría muchas vidas que se pierden en la carretera.

1.2. Descripción del problema

El grupo de investigación *CAOS* (Control, Aprendizaje y Optimización de Sistemas) de la Universidad Carlos III de Madrid está desarrollando un sistema de detección de patrones de comportamiento en los conductores de un vehículo. La implementación de cámaras y sensores en un vehículo real es una tarea sumamente costosa, además de lo peligroso que supone experimentar conduciendo un coche real. Por ello, el grupo *CAOS* ha adquirido un simulador de coche denominado *STISIM* [5] *(ver* Figura 2*)*, el cual permite simular en tiempo real una conducción normal de una persona, incluyendo sensores simulados en el propio simulador.

Figura 2: Entorno de simulación - Simulador de coche

Este simulador cuenta con una gran cantidad de sensores que permiten registrar toda la información generada durante un proceso de conducción. En estos datos se encuentran, entre otros, el ángulo de giro de volante en cada instante de tiempo, el nivel en el cual se están pisando los pedales de acelerador, embrague y freno, a qué distancia se encuentra el vehículo del vehículo delantero o trasero, etc.

Además, el grupo también cuenta con una cámara Kinect XBOX360 (*ver figura 3)*, la cual permite la detección de la posición de la cara y los ojos del conductor mediante un algoritmo desarrollado por el grupo de Sistemas Inteligentes del departamento de automática [6].

Figura 3: Kinect XBOX 360 [7]

El problema se basa en estudiar y desarrollar un modelo teórico-práctico de actividades que defina las diferentes actividades que se puede realizar mientras se conduce un vehículo, sentando las bases de un primer modelo práctico que sea capaz de detectar, en un futuro, patrones de comportamiento de un conductor en tiempo real, haciendo uso de diferentes sensores y cámaras. Además, es necesario obtener los diferentes datos de la cámara y sensores que permitan realizar el modelo.

1.3. Alcance del proyecto

El objetivo principal de este proyecto es desarrollar un modelo teórico-práctico de actividades, que siente las bases de un modelo práctico que permita reconocer patrones de comportamiento en tiempo real.

Los objetivos específicos del presente proyecto son:

- Desarrollar teóricamente un modelo ideal de actividades de conducción.
- Obtener diferentes datos de sensores simulados y cámara que se ajusten a los datos definidos en el modelo teórico, almacenando dichos datos en ficheros.
- Implementar un programa que procese, transforme y razone con los datos obtenidos.
- Elaborar una primera aproximación de un modelo práctico que reconozca alguna actividad propuesta.

1.4. Organización de la memoria

En el *Capítulo 1* se recoge la introducción al presente proyecto, haciendo hincapié en la motivación de realizarlo, en la descripción del problema y en los objetivos generales y

específicos del proyecto. Por último, se define la organización de la memoria del presente documento.

En el *Capítulo 2* se desarrolla el estado del arte, es decir, se expone y desarrolla el estado actual del reconocimiento de actividades por visión y sensores en el ámbito de la investigación. Seguidamente, se desarrollan los sistemas ADAS (*Advanced Driver Assistance Systems*, Sistemas Avanzados de Ayuda a la Conducción). Por último, se describen las principales aplicaciones que tiene el reconocimiento de actividades en la sociedad.

En el *Capítulo 3* se desarrolla un modelo teórico de actividades ideales en el campo de la conducción. Primeramente, se estudia la viabilidad de los datos, es decir, qué datos son factibles de extraer de la cámara y los sensores. Finalmente, se propone un modelo teórico de actividades basado en jerarquías, desarrollando teóricamente cada una de las actividades propuestas.

En el *Capítulo 4* se documenta el análisis y diseño del sistema a implementar, el cual es capaz de procesar, transformar y razonar con los datos generados por la cámara y los sensores. Primeramente, se establecen las normas y restricciones del sistema. Seguidamente se describe el entorno operacional sobre el que se realiza el presente proyecto. Posteriormente se especifican los requisitos y casos de uso del sistema. Por último, se define la arquitectura (diseño) del sistema a implementar.

En el *Capítulo 5* se desarrolla la implementación y experimentación de sistema, describiendo con sumo detalle el procesado, transformación y razonamiento de los datos generados. Además, se incluye un *diagrama UML* que muestra las relaciones existentes entre las diferentes clases, métodos y variables que componen la codificación del sistema. Por último, se detalla la experimentación llevada a cabo.

En el *Capítulo 6* se define la planificación y presupuestos del presente proyecto, realizando un *diagrama de Gant*t para la planificación y un estudio de costes materiales y de personal para los presupuestos del presente proyecto.

En el *Capítulo 7* se definen las conclusiones del presente proyecto, así como los trabajos futuros que se podrían realizar partiendo de este proyecto.

Para finalizar el documento, se encuentra un glosario de términos, donde se pueden encontrar las definiciones de palabras clave y técnicas del documento, las referencias, donde se encuentran las fuentes bibliográficas que se han seguido para el desarrollo del proyecto, y los anexos al documento, incluyendo manual de usuario, resumen en lengua inglesa y otro tipo de información relevante.

Capítulo 2: Estado del Arte

2. Estado del arte

En este capítulo de la memoria se realiza una revisión bibliográfica del estado del arte en el área del reconocimiento de actividades, así como una revisión de los principales modelos y técnicas estudiadas y desarrolladas para este fin.

Poder conocer que actividad está realizando una persona, un robot o cualquier otro tipo de sistema es una tarea compleja que permite obtener información relevante y útil, aplicable a numerosos campos de la sociedad (ver 2.4. Aplicaciones).

El reconocimiento de actividades está relacionado con numerosas áreas de investigación, por lo que existen innumerables estudios de diferentes investigadores a la hora de definir y describir el proceso del reconocimiento de actividades. El presente estudio bibliográfico se centra en el reconocimiento de actividades mediante sensores y visión por computador, ya que son los dos enfoques que permiten cumplir con los objetivos del presente proyecto.

Tal y como se detalla en [8], existen numerosos enfoques para el reconocimiento de actividades, siendo los más destacables el reconocimiento de actividades basado en visión y el reconocimiento de actividades basado en sensores. Es muy común combinar todos estos enfoques para llegar al objetivo final: el reconocimiento de la actividad.

Dentro del área del reconocimiento de actividades es importante definir los actores o componentes implicados en esta tarea [8]:

- **Sujeto reconocido**: sujeto que realiza la actividad que se desea reconocer.
- **Sujeto reconocedor:** sujeto que lleva a cabo el reconocimiento. Suele ser un sistema informático o un agente software.
- **Datos:** estos datos sirven como base para construir un modelo capaz de reconocer una actividad. Los datos pueden ser extraídos mediante sensores, cámaras o cualquier otro tipo de sistema.
- **Modelo:** a partir de los datos generados, se generan modelos capaces de reconocer actividades mediante numerosas técnicas, entre las que destacan la minería de datos (*data mining)*, aprendizaje automático y técnicas estadísticas.

2.1. Reconocimiento de actividades basado en visión

El reconocimiento de actividades humanas es una disciplina muy activa en la visión por computador, cuyo objetivo es clasificar y reconocer automáticamente acciones o actividades humanas a partir de los fotogramas que componen una secuencia de vídeo, utilizando fundamentalmente algoritmos de reconocimiento de patrones [2].

El reconocimiento de actividades basado en visión utiliza las cámaras de vídeo para poder grabar una secuencia de vídeo, la cual posteriormente es analizada y procesada. Este enfoque es bastante interesante, puesto que se emplea en áreas tan importantes como la

video vigilancia, pudiendo reconocer, por ejemplo, si un sujeto desconocido está intentando acceder a una vivienda ajena.

Para poder reconocer una acción o una actividad mediante videocámaras (reconocimiento de actividad basado en visión por computador) es necesario reconocer individualmente cada una de las imágenes o fotogramas presentes en una secuencia de video [9]. Históricamente, el reconocimiento visual de imágenes humanas se ha centrado en el reconocimiento de gestos, expresiones faciales, movimientos de los pies, etc. Sin embargo, actualmente, se pretende reconocer los movimientos de todo el cuerpo.

Existen varios enfoques acerca del reconocimiento de actividades mediante visión. La mayoría de ellos asumen que este proceso se divide en varias tareas. Habitualmente, este proceso se divide en [9]:

- **Extracción de características (datos)**: es la principal tarea del reconocimiento de acciones y consiste en extraer la forma, postura, gestos y señales de movimiento de video que son discriminatorias respecto a otras acciones. Estas características extraídas deben de ser lo más discriminatorias posibles. Es posible dividir estas características en:
    - Espaciales: características dentro de un mismo fotograma.
    - Temporales: características de un subconjunto de fotogramas obtenidos en el tiempo.
    - Espacio-temporales: combinación de dos anteriores
    - Locales: extraen puntos de interés. Las funciones de extracción de características locales encuentran zonas o regiones donde la información visual cambia (puntos de interés).

- **Aprendizaje y clasificación de acciones:** en estas tareas se aprenden modelos estadísticos de las características extraídas, usando dichos modelos para clasificar nuevas observaciones.

- **Segmentación de acciones:** esta tarea es necesaria para segmentar o cortar trozos de movimientos en acciones primitivas que sean consistentes con el conjunto inicial de secuencias de entrenamiento

Otros trabajos, como [2], asumen que el proceso de reconocimiento de actividades por visión puede dividirse en (ver *figura 4)*:

- Pre-procesamiento del video
- Representación de información visual
- Aprendizaje automático
- Clasificación

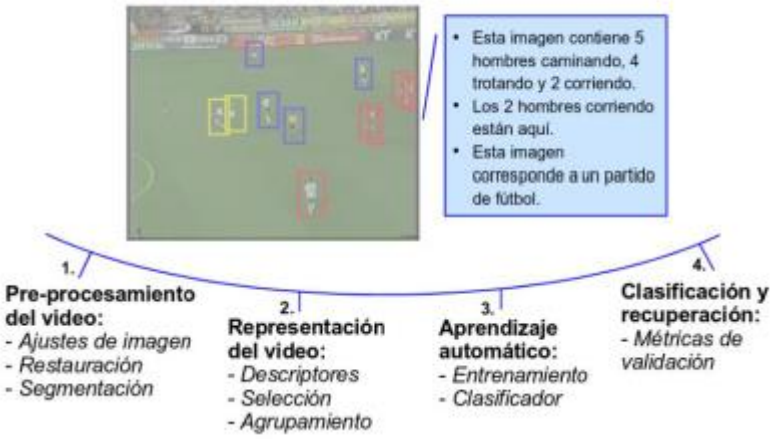

Figura 4: Proceso de reconocimiento de actividades [2]

Tras haber analizado los diferentes modelos, se estudia con más detalle el primer enfoque descrito, ya que se considera que es el que más aplica al presente proyecto.

Las técnicas basadas en visión por computador para representar, segmentar y reconocer acciones humanas, de acuerdo a [9], pueden ser clasificadas en base a diferentes criterios, como por ejemplo, qué partes del cuerpo humano están involucradas, en base a las características de las imágenes seleccionadas para la discriminación (regiones y puntos de interés), o en base a los modelos utilizados para el aprendizaje, como pueden ser *modelos de Markov (HMM) o kNN.*

La mayor parte de trabajos o estudios, según [9], reflejan que la mejor clasificación o taxonomía de estas técnicas es en base a como tratan la **estructura espacio-temporal de las acciones.**

2.1.1. Técnicas para representar y reconocer acciones en base a su estructura espacio-temporal

Existen una gran cantidad de técnicas para representar y reconocer actividades o acciones en base a la estructura espacio-temporal de dichas acciones. Tal y como se detalla en [10], un vídeo que contiene acciones humanas involucra a personas interrelacionadas y contiene unas estructuras espacio-temporales muy destacables. En otras palabras, las acciones primitivas que componen una actividad presentan unas fuertes relaciones espacio-temporales.

A continuación se presentan las principales técnicas utilizadas para la extracción y representación de características de las acciones en base a su estructura espacio-temporal.

2.1.1.1. Estructura espacial

La estructura espacial de una acción hace referencia a las características locales de las secuencias de vídeo (de los fotogramas), sin tener en cuenta la duración de las acciones ni el orden que presentan.

El primer paso del reconocimiento de acciones es extraer aquellas características de las imágenes que son discriminatorias respecto a la postura y el movimiento del cuerpo humano. Para seleccionar los datos (características) que son extraídos es necesario estudiar la información que representan y la complejidad.

Las características a extraer de la estructura espacial de una actividad pueden obtenerse en base a tres modelos: modelos de cuerpo, modelos de imágenes y modelos estadísticos.

MODELOS DE CUERPO

En esta sección, se estudian métodos que representan la estructura espacial de las acciones referidas al cuerpo humano. En cada *frame* observado, se recupera la posición del cuerpo. El reconocimiento de acciones es realizado en base a estimaciones de la pose humana.

Johansson [11] demostró que los humanos pueden reconocer acciones simplemente con el movimiento de unos puntos de luz (ver Figura 5: *Moving Light Displays* [11]*)* adjuntos al cuerpo humano (*Moving Light Displays o MLD)*.

Figura 5: *Moving Light Displays* [11]

Sus experimentos, durante décadas, estuvieron basados en el movimiento de dichos puntos. También fueron originados de la discusión no resuelta de si los humanos reconocen acciones directamente de patrones de movimiento en 2D o de si ellos computan una reconstrucción 3D del movimiento de patrones.

MODELOS DE IMÁGENES

En este modelo no se detectan partes de cuerpo. En lugar de eso, se detecta una región de interés centrada alrededor de la persona (*Region of Interest* o ROI).

Los modelos de imágenes son más robustos, simples y eficientes que los modelos de cuerpo. Además, se ha demostrado que son tan discriminativos como los modelos de cuerpo.

Darell [12] presentó un modelo donde las imágenes de los gestos de las manos están directamente correlacionadas. Otros modelos importantes son los siguientes:

- Modelos de imágenes que usan siluetas y contornos de agentes humanos que están realizando acciones.
- Modelo de clases de imágenes usan la densidad del flujo óptico.
- Modelo basado en gradientes.
- Modelo inspirado en el enfoque HMAX.

MODELO ESTADÍSTICO

Las imágenes se descomponen en regiones muy pequeñas (dichas regiones no están enlazadas con partes del cuerpo). Las acciones son reconocidas en base a estadísticas de las características locales de todas las regiones. Una ventaja de ello es que no dependen del etiquetado explícito de las partes del cuerpo ni de la detección y localización explícita del humano.

Se introducen los llamados puntos interesantes espacio-tiempo y son específicamente introducidos para generar puntos de interés. Este enfoque está basado en la estrategia *bottom-up*, donde primero se determinan los puntos interesantes de las imágenes y luego se asigna cada región a un conjunto preestablecido de características. De esta manera, la clasificación de imágenes se reduce a computar dichas características.

El modelo más conocido de los modelos estadísticos es el modelo *Bag Of Features.*

BAG OF FEATURES

Este modelo se basa en representar una imagen como un vector de características (ver Figura 6: Esquema BOF [2]). A continuación se describen los principales pasos seguidos:

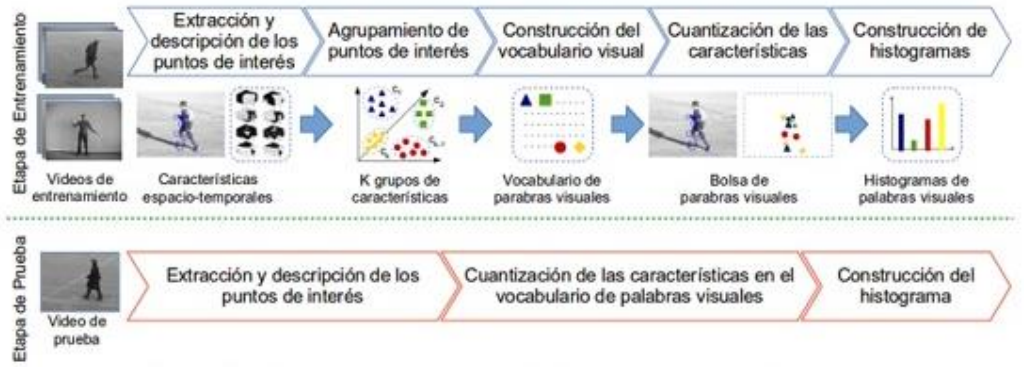

Figura 6: Esquema BOF [2]

- *Extracción y agrupamiento de características espacio-temporales*: Existen varios modelos para la extracción de características (ver Figura 7), como puede ser el modelo basado en una cuadrícula regular [13]. Por otro lado, para agrupar las características se utiliza el algoritmo *K-means* u otro método no supervisado.

Figura 7: Extracción de características. BOF. [13]

- *Construcción y aprendizaje de un corpus o vocabulario visual:* Es necesario generar un vocabulario visual (ver Figura 8) que no sea ni demasiado amplio (palabras innecesarias, ruido) ni demasiado escaso (falta de información). Determinar cuál es el tamaño óptimo de dicho vocabulario es una tarea que aún no ha sido resuelta (demasiado compleja). El tamaño óptimo es aquel en el que cada palabra represente un concepto único.

Figura 8: Generar corpus. BOF. [13]

- *Cuantificación de características utilizando el corpus (asignación de términos):* Dada una nueva imagen, se extraen las características de dicha imagen y se asignan, por ejemplo, mediante vecinos más cercanos a los términos del vocabulario.

- *Generación vector de términos y construcción de histograma:* se crea una vector de términos (histograma) en el que se representa la importancia de cada término respecto al vocabulario visual generado (ver Figura 9).

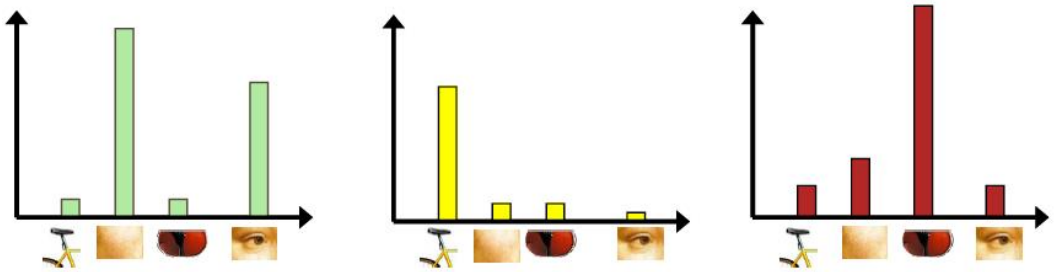

Figura 9: Histograma del BOF. [13]

2.1.1.2. Estructura temporal

La estructura temporal de una acción hace referencia a que una acción tiene un comienzo y un final (acciones durativas) y, además, las acciones pueden seguir un determinado orden secuencial que las hace más fácil de reconocer [14]. La información temporal se puede codificar en tres tipos según el modelo presentado en [10]: concurrencia (acciones que suceden al mismo tiempo), anterior (una acción ocurre antes que otra) y posterior (una acción ocurre después de otra).

El modelado temporal de acciones humanas en términos de primitivas a bajo nivel tiene una amplia historia en las investigaciones de visión por computador. Según se detalla en [10], Yamato usó modelos ocultos de Markov (HMMs) para descubrir la evolución temporal de acciones. Por otro lado, Moore y Essa construyeron gramáticas estocásticas para representar los componentes de las acciones.

En la sección anterior se ha estudiado los diferentes tipos de características de imágenes que se pueden extraer de un vídeo para representar su estructura espacial. A continuación, se detallan las principales y diferentes técnicas que pueden ser utilizadas para aprender la estructura temporal de las acciones [9].

GRAMÁTICAS DE ACCIONES

En la estructura temporal, una acción se puede definir como una secuencia de momentos. Una de las maneras o formas de desarrollar un sistema capaz de reconocer acciones en base a la observación de características temporales es agrupar dichas características dentro de estados y aprender funciones de transición entre ellos. Para realizar este procedimiento se suelen emplear las gramáticas probabilísticas.

Una gramática probabilística es una gramática libre de contexto en la cual cada regla tiene asignada una probabilidad (probabilidad entre transición de estados) [15]. Las gramáticas probabilísticas más utilizadas en el campo del reconocimiento visual de acciones son los modelos ocultos de Markov (Figura 10) y los modelos CRF (modelo de Markov independiente).

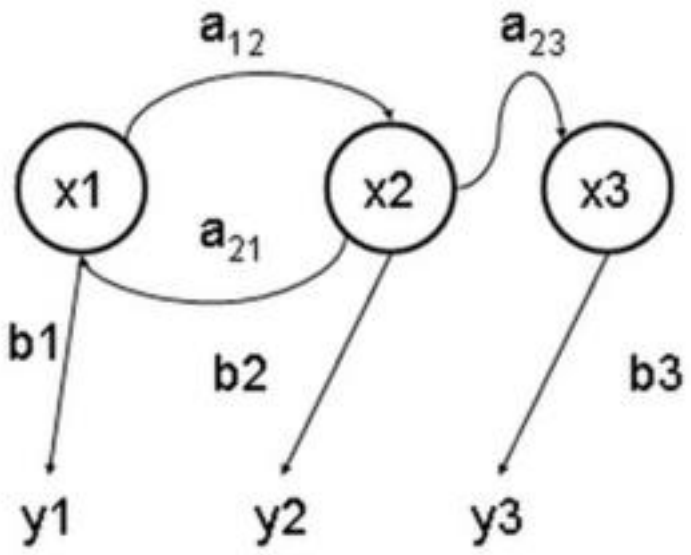

Figura 10: HMMs [wikipedia.org]

En un *HMMs*, los estados (x1,x2,..,xN) son ocultos, mientras que las salidas (y1,y2…) son parcialmente observables. A partir de una probabilidad de transición y de emisión se trata de determinar en qué estado se encuentra el modelo en el instante "t".

La principal limitación de estos modelos es que son modelos de acciones puramente secuenciales, con amplias limitaciones para el reconocimiento de acciones de todo el cuerpo donde las diferentes partes del cuerpo se mueven en paralelo.

Los modelos CRF son modelos ocultos de Markov discriminatorios que pueden usar características no independientes y observaciones a lo largo del tiempo, contrariamente a los modelos ocultos de Markov.

La principal ventaja de las gramáticas de acciones es su alto grado de modularidad [4], es decir, la capacidad de ser vistas como un conjunto de módulos.

PLANTILLAS

En el ámbito de la visión por computador [9], una plantilla es una secuencia de fotogramas durante un determinado tiempo que representan una acción (Figura 11). Los métodos basados en plantillas tratan de aprender bloques temporales de características que representan una acción. Las plantillas son vistas como representaciones vectoriales de tamaño fijo.

Se diferencian de las gramáticas probabilísticas en que las plantillas no pueden representar variaciones en el tiempo ni velocidad. En este caso, las variaciones son representadas de manera implícita a través de grandes conjuntos de bloques.

Todos los vectores de características (plantillas) deben de tener la misma dimensión, por lo que es necesario normalizarlos.

La principal desventaja de este modelo es la mala generalización cuando los datos están incompletos.

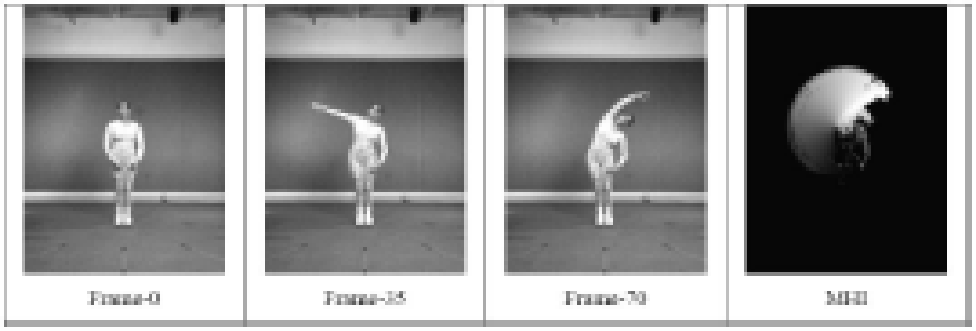

Figura 11: Plantillas de acciones [9]

ESTADÍSTICAS TEMPORALES

Uno de los modelos más utilizados para reconocer acciones en base a las estadísticas temporales es el modelo "*key-frame*" [9]. Dicho modelo se basa en determinar cuál es la característica única que permite reconocer una acción.

Otros de los modelos típicos es el modelo "*bag of features*" (modelo explicado en estructura espacial). En este caso, la frecuencia de las características está representada en función del tiempo.

2.1.2. Reconocimiento de acciones primitivas basado en la localización de acciones

Tal y como se describe en [10], reconocer una acción en un espacio y en un tiempo determinado es seguramente un paso crucial en el reconocimiento de actividades. El estudio de [10] se basa en el reconocimiento de acciones en vídeo en base a la localización en espacio y tiempo de las acciones primitivas. Dicho estudio pretende reconocer acciones primitivas en un conjunto de entrenamiento que contiene secuencias de vídeo. Este trabajo sigue la premisa de que las acciones de una misma secuencia de vídeo tienden a tener una gran correlación, ya que cada acción se produce a continuación de otra, en un orden temporal (estructura temporal). Se propone un esquema jerárquico de *clustering* (Figura 12) para reconocer acciones primitivas de los vídeos.

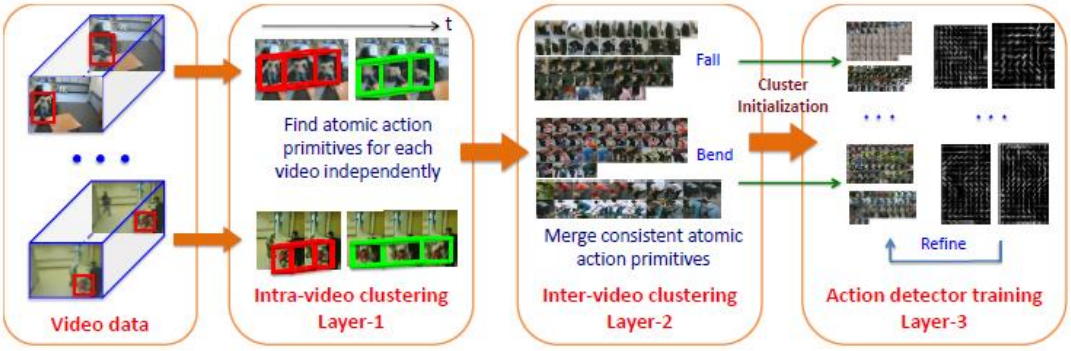

Figura 12 : Modelo de reconocimiento de actividades en base a la localización en espacio y tiempo de las acciones [10]

En la primera capa del modelo, se pretende encontrar la similitud entre cada vídeo. Las acciones contenidas en un mismo vídeo forman múltiples "clusters espacio-temporales". Para la similitud espacial, se usa la distancia euclídea entre dos ejemplos, mientras que para la similitud temporal se utiliza el número de *frames* entre cada vídeo. En definitiva, esta primera capa agrupa las acciones dentro de un mismo vídeo.

Una vez obtenidos una colección de *clusters* en los que cada *cluster* contiene ejemplos de un mismo vídeo, es necesario unir *clusters* de diferentes vídeos, es decir, unir aquellos ejemplos que representen la misma acción en un mismo clúster. Esto se realiza en la segunda capa. Para cada clúster, se entrena una máquina de soporte vectorial (SVM) para realizar esta tarea.

Por último, en la tercera capa, se pretende entrenar un detector de acciones que permita detectar automática y simultáneamente las acciones y las personas que realizan estas acciones. Este detector de acciones está basado en las máquinas de soporte vectorial (SVM).

2.2. Reconocimiento de actividades basado en sensores

El reconocimiento de actividades basado en sensores utiliza sensores situados estratégicamente. Sin embargo, elegir qué sensores utilizar no es una tarea sencilla. En [14] se lleva a cabo un estudio de qué sensores son útiles en un vehículo. Los sensores son agrupados en conjuntos y son colocados a partir algoritmos que determinan dónde es recomendable situarlos. Los sensores son utilizados para la recolección de datos del vehículo. Se estudian dos enfoques claramente diferenciados:

- Equipar una flota de vehículos con sensores: es lo más natural posible. Sin embargo, es muy costoso.
- Equipar sensores avanzados virtuales en un simulador de conducción: permite simular cualquier situación en la carretera, además de ser menos costosos

Evidentemente, para este estudio, se elige equipar los sensores virtuales en el simulador, ya que la implementación de sensores reales es sumamente costoso y peligroso.

Finalmente, mediante el algoritmo *Random Forest*, se demuestran que los sensores más útiles son aquellos que detectan la posición de cara y ojos (sistema de reconocimiento visual mediante cámara interna) y aquellos que detectan la aceleración o deceleración del coche.

Por otro lado, según se detalla en [8], se puede establecer una taxonomía de los sensores en base a su interacción con el ser humano en el ámbito de reconocimiento de actividades. En este enfoque, los sensores se clasifican en:

- **Sensores portátiles**: esta clase de sensores son aquellos que se adhieren, directa o indirectamente, al cuerpo humano. En la actualidad, el desarrollo de nuevas tecnologías como *smartphones* o *smartwatches* permiten que las personas estén localizadas constantemente mediante GPS. En los últimos dos años, además, se han desarrollado pulseras de actividad que permiten registrar datos como las horas de sueño o la actividad física desarrollada.

- **Sensores fijos**: estos sensores son aquellos que están situados en objetos con los que el ser humano interactúa. En el ámbito de la automoción, un sensor colocado en el acelerador que indique con qué intensidad se está pisando sería un ejemplo típico de este tipo de sensores.

Una vez estudiado la taxonomía de los sensores, es necesario determinar los principales pasos a seguir para reconocer una actividad mediante el uso de sensores. Lo primero a realizar es elegir qué sensores se van a utilizar y donde se van a situar. Seguidamente, se debe recolectar, procesar y almacenar los datos proporcionados por los sensores. Por último, se debe desarrollar un algoritmo que permita construir un modelo que determine qué actividad se está realizando. En concreto, el proceso para reconocer una actividad mediante sensores [8] se basa en:

El proceso para reconocer una actividad mediante sensores [5] se basa en:

- **Elección y despliegue de sensores:** es necesario elegir qué sensores serán más útiles para el reconocimiento de actividades. Una vez seleccionados, se despliegan en objetos y en el entorno para registrar el comportamiento de un usuario en un determinado entorno. En el caso de la automoción, se instalan sensores en el vehículo y se monitoriza la interacción del usuario con dichos sensores.

- **Recolección y almacenamiento de datos:** mediante técnicas de análisis de datos, los datos son recogidos, almacenados y procesados.

- **Desarrollo de algoritmos:** se desarrollan algoritmos (o se seleccionan algunos ya desarrollados) que permitan razonar con los datos de la fase anterior, con el objetivo de inferir la actividad que está realizando el usuario.

2.3. Taxonomía de actividades

Debido a la variedad de complejidad de las diferentes actividades humanas, así como a la diferencia de abstracción que presentan cada una de ellas, es necesario establecer una taxonomía de éstas. Existen diferentes modelos [2] que realizan una taxonomía de las actividades:

- ***Modelo de Moeslund y colaboradores***: los movimientos se pueden dividir en primitiva (entidades atómicas a partir de las cuales se construyen las acciones), acción (entidades a partir de las cuales se construyen las actividades) y actividad. Debido a esta definición, la jerarquía de las acciones depende de la aplicación. Por ejemplo, en robótica una acción primitiva es aquella que, combinada con otras, generan un movimiento en el robot. En el tenis, es el golpe de derecha o el revés, por ejemplo. En cambio, una acción es la secuencia de primitivas para llegar a ella. En el tenis, sería el conjunto de primitivas para devolver una pelota: posicionarse, mirar la pelota, mover el brazo y golpear. La actividad, dentro del contexto del tenis, sería jugar al tenis. Jugar al tenis está compuesto de múltiples acciones, como pueden ser devolver la bola, realizar un saque, recoger la pelota, etc.

- ***Modelo de Chaararoui***: jerarquía de cuatro niveles en base al grado semántico y duración del movimiento (ver Figura 13). Cuanto más arriba se encuentra la acción en el modelo jerárquico, mayor será su complejidad y su tiempo de realización

Figura 13: Jerarquía de acciones [2]

A continuación se define cada uno de los niveles de la pirámide jerárquica del modelo propuesto en [2], el cual parte del modelo propuesto por Chaararoui:

Clasificación	Descripción	Intervalo de tiempo	Ejemplo
Movimiento	Acción primitiva	fotograma	Mover una pierna
Acción	Conjunto o repetición de primitivas que semántica son un todo	Segundos o minutos	Caminar
Acción simple	Ejecutada por un individuo	Segundos o minutos	Caminar
Acción compuesta	Ejecutada por más de un individuo	Segundos o minutos	Besar
Actividad	Secuencia de acciones en orden	Segundos o minutos	Cocinar
Situación	Actividad que adquiere significado en base al contexto	Segundos o minutos	Pedir ayuda o pedir un pase de fútbol
Evento	Combinación de actividades en las que están involucrados varios individuos	Minutos u horas	Jugar al fútbol
Comportamiento	Combinación de todos los anteriores	Horas, días, semanas, meses, años	Modo o estilo de vida

Tabla 1: Modelo de Chaararoui

2.4. Aplicaciones

El reconocimiento de actividades mediante visión por computador y sensores tiene numerosas aplicaciones en diferentes campos. Se pueden agrupar, de acuerdo a [8] y [16], en:

- **Seguridad y vigilancia:** en este ámbito destacan las cámaras de video vigilancia que permiten reconocer actividades sospechosas como puede ser la de dos individuos peleándose o entrando a la fuerza en algún establecimiento público o privado.

- **Entretenimiento:** cada vez son más comunes sistemas que reconocen automáticamente jugadas determinadas en eventos deportivos (ver Figura 14).

Figura 14: Reconocimiento de jugadores [16]

- **Salud:** destacan los sensores que registran el pulso cardiaco de los pacientes. Por otro lado, también hay numerosos sistemas que ayudan a pacientes con problemas psicomotrices.

- **Militar:** reconocimiento de tácticas militares o monitorización del estado de salud de soldados.

2.5. Sistemas ADAS

Los sistemas *ADAS* (*Advanced driver assistance systems)* [17] están a la orden del día en la actualidad.. A continuación, se exponen algunos de los principales sistemas de asistencia a la conducción:

- **Control de crucero adaptativo:** sistema que permite mantener una distancia de seguridad razonable con el vehículo delantero (ver Figura 15). Este sistema utiliza tecnología láser para detectar el vehículo delantero.

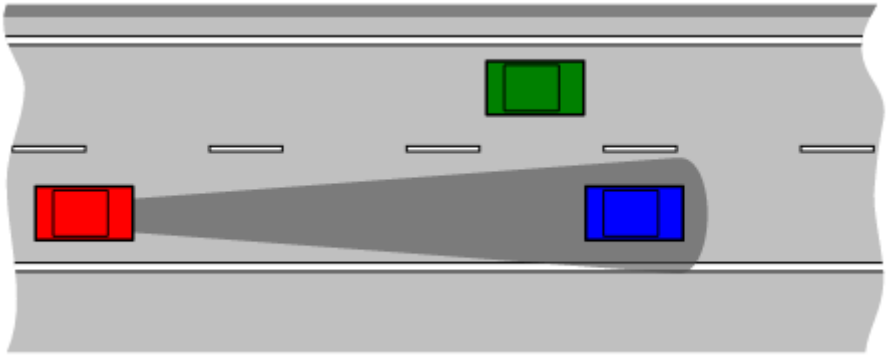

Figura 15: Control de crucero [17]

- **Aparcamiento automático:** sistema que permite aparcar de manera automática un vehículo en un hueco libre (ver Figura 16). Se basan en sensores de distancia y ultrasonidos que permiten medir el hueco libre, así como la distancia lateral a los vehículos ya aparcados. Además, algunos de estos sistemas incorporan cámaras de vídeo para que el usuario revise con seguridad la maniobra automática.

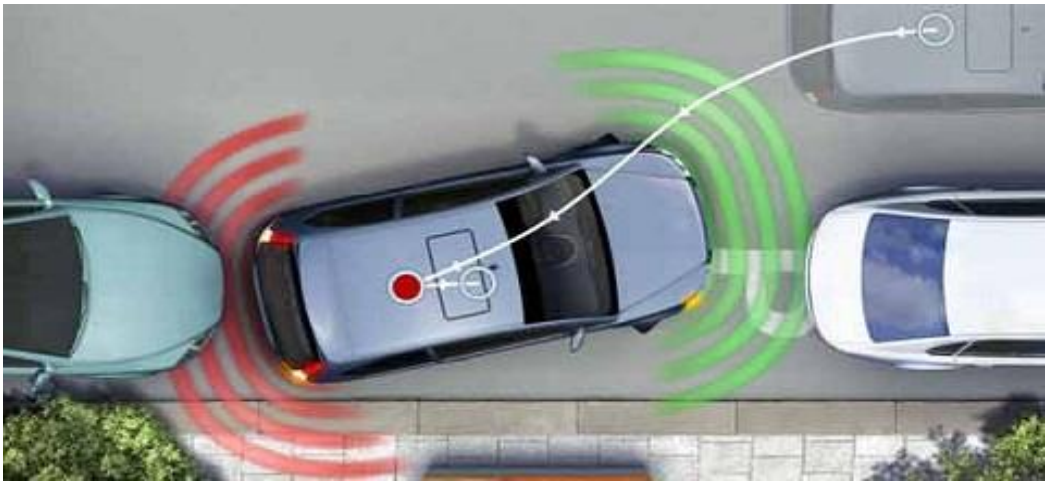

Figura 16: Aparcamiento automático [18]

- **Aviso de cambio de carril:** sistema que avisa al usuario de un cambio de carril involuntario, es decir, cambio de carril en el que el intermitente no estaba activo (ver Figura 17). Esta situación se puede producir si el conductor se duerme. Para implementar este sistema, se utilizan sensores de infrarrojos y cámaras de vídeo.

Figura 17: Cambio de carril involuntario [19]

- **Reconocimiento automático de señales:** una cámara situada generalmente en el interior del parabrisas permite detectar de manera automática todas las señales, visualizándose éstas en el navegador del vehículo para que el usuario pueda verlas correctamente (ver Figura 18).

Figura 18: Reconocimiento de señales [20]

Capítulo 3: Modelo de Actividades Ideales

3. Definición de modelo de actividades ideales

En este capítulo se define un modelo de actividades ideales que realiza un conductor, estableciendo una taxonomía de estas actividades, siendo esta una de las tareas más importante del reconocimiento de actividades. Una definición incorrecta de las actividades llevaría a la generación de unos conjuntos de datos que no se corresponderían con las actividades que realiza habitualmente un conductor (conjuntos de test) y, por lo tanto, sería muy difícil realizar un modelo práctico óptimo.

Una vez se definen las actividades ideales, el investigador realiza dichas actividades en el simulador para generar los ficheros *logs*, que posteriormente servirán como conjuntos de entrenamiento en el proceso de detección automática de actividades.

Estos ficheros de datos servirán como modelos de actividades ideales. Posteriormente, será posible detectar actividades en tiempo real comparando los *logs* generados en ese instante con los *logs* de las actividades ideales.

El primer paso en la definición de actividades ideales es estudiar qué datos (variables) se pueden obtener, tanto de un simulador de coche como de la cámara, es decir, estudiar de qué sensores se dispone para la recolección de datos. No es recomendable definir actividades que contengan datos que no se puede extraer.

Seguidamente, se propone un modelo jerárquico de actividades humanas, estudiando qué solución es la más apropiada entre las propuestas.

Posteriormente, se definen los eventos atómicos, acciones, tareas y maniobras definidos en el modelo propuesto.

3.1. Viabilidad de datos

Como se comenta anteriormente, en este apartado se estudia qué datos se pueden obtener. El objetivo de conocer estos datos es saber qué datos se disponen de antemano para la definición de actividades de conducción. Las dos fuentes de datos de las que se disponen son el simulador de coche y la cámara Kinect.

3.1.1. Datos del simulador

Para los datos del simulador, se asume que todos los datos necesarios se pueden obtener de él, ya que los simuladores de conducción están diseñados, entre otras cosas, para ello. Para conocer de manera específica qué datos se disponen, se realiza una lectura del manual correspondiente.

Tras realizar una lectura de uno de los manuales del simulador [18], se concluye que el simulador proporciona la mayoría de los datos necesarios para la definición de estas actividades y su posterior implementación práctica.

En caso de usar otro software de simulador, como puede ser el STISIM M300 o STISIM M500, será necesario leer su manual correspondiente.

Es necesario destacar que para cada actividad se necesitan datos diferentes. En la actividad de adelantar, por ejemplo, es necesario obtener la velocidad del coche que se quiere adelantar, así como la distancia del vehículo del conductor al vehículo que se desea adelantar.

A pesar de lo expuesto anteriormente, se asume que existen una serie de variables que serán comunes a todas las actividades, ya que representan los datos básicos de un vehículo (velocidad, nivel de freno, revoluciones, etc.). Estos datos se pueden extraer de manera directa del simulador, sin necesidad de inferirlos de otros datos.

A continuación se expone de manera tabular los datos más importantes que se pueden extraer del simulador que serán comunes a todas las actividades, así como sus rangos de valores y descripción.

DATOS SIMULADOR		
Variables	**Rango devuelto**	**Descripción**
Speed	[0, infinito]	Velocidad (km/h)
Throttle	[0,65535], siendo "0" pisar máximo	Nivel de pisar acelerador
Braking	[0,65535], siendo "0" pisar máximo	Nivel de pisar freno
Gear	[0,7], con "7" marcha atrás	Marcha actual
revolutions	[0, máximo rpm vehículo]	RPM
steeringwheelAngle(SWA)	[0, 65535] siendo "0" lo más a la izq.	Ángulo del volante
clutched	[0,65535], siendo "0" pisar máximo	Nivel de pisar embrague

Tabla 2: Datos básicos simulador

3.1.2. Datos cámara

La segunda fuente de datos que se utiliza en el sistema es la cámara Kinect, la cual mediante una aplicación desarrollada por el departamento de automática [6], devuelve, para cada instante de tiempo, un vector de la posición de la cara y un vector de la posición de los ojos. Partiendo de esa aplicación, se han hecho modificaciones en el código para que devuelva posiciones absolutas y alfabéticas de la cara y ojos.

El vector de la cara está formado por tres valores: movimiento horizontal, vertical y lateral de la cara(*pitch, roll y yaw)*. Es necesario discretizar esos valores numéricos en valores alfabéticos que indiquen la posición de la cara a partir de ese vector. Primeramente, se discretiza cada uno de los tres valores del vector y, posteriormente, se deduce una posición de la cara como combinación de estos valores discretizados.

En cuanto al vector de los ojos, la aplicación devuelve un “0” si los ojos miran hacia la derecha, un “1” si miran hacia el centro y un “2” si miran hacia la izquierda, por lo que la discetización es inmediata.

Por consiguiente, se dispone de los siguientes datos (variables):

DATOS CÁMARA	
variable	**valores**
headPosition	[fronthead, front-right, front-left, front-up,front-down, up-right, up-left, down-right, down-left]
eyesPosition	[right, center, left]

Tabla 3: Datos cámara

3.2. Modelo jerárquico de actividades

En este apartado se estudian los principales modelos taxonómicos de acciones. Estos modelos han sido descritos en el apartado 2.3. Taxonomía de actividades. Tras analizarlos detalladamente, se concluye que estos modelos parten de la misma idea: definen la taxonomía de acciones en base a la duración, grado semántico y la complejidad de las acciones.

Se parte de esta conclusión para elaborar un modelo propio de acciones basado en estos modelos, realizando un modelo jerárquico con diferentes niveles de complejidad de acciones. Este modelo aprovecha todos los conceptos estudiados en los diferentes modelos para establecer la mejor taxonomía posible. A esta misma conclusión se llega en [2], donde los autores proponen un modelo basado en los aspectos en común del resto de modelos.

La taxonomía propuesta (*ver* Figura 19*)* consta de diferentes niveles jerárquicos en forma piramidal. Cada nivel superior indica mayor nivel de complejidad, mayor nivel semántico y mayor tiempo de realización de la acción. Además, las acciones de niveles superiores están formadas por acciones de niveles inferiores.

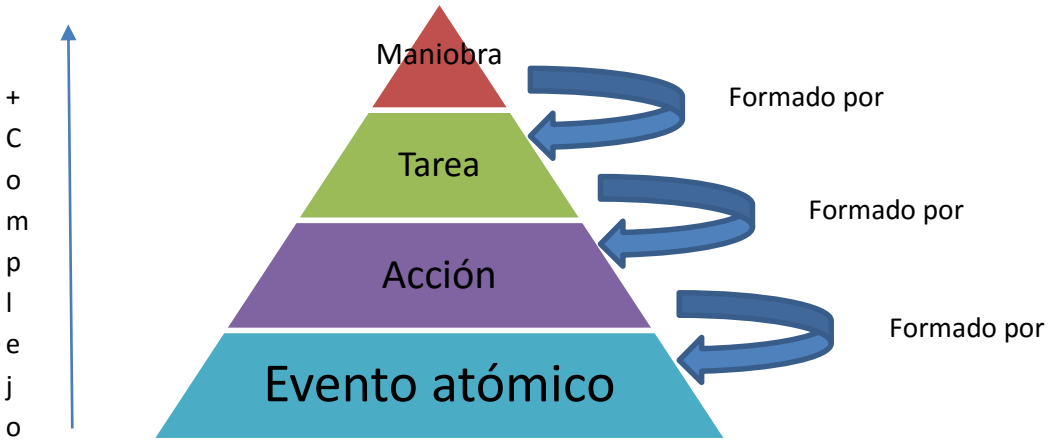

Figura 19: Modelo jerárquico de actividades

Como se puede ver, el modelo propuesto consta de cuatro tipos de actividades: maniobra, tarea, acción y evento atómico. De esta manera, cada actividad de un nivel superior está compuesto por actividades del nivel inferior. De este modo, por ejemplo, la maniobra de "Adelantar" puede estar formada por dos tareas. Estas dos tareas a su vez pueden estar formadas por cuatro acciones. Por último, cada acción puede estar formada por varios eventos atómicos.

Es importante destacar que una actividad de un nivel inferior puede formar parte de una actividad de un nivel superior sin formar parte de un nivel intermedio entre éstos. De esta manera, la maniobra de adelantar puede estar formada por dos "tareas" y tres "eventos atómicos", sin que éstos últimos formen alguna "acción". Que una actividad se encuentre en un nivel superior indica que es más compleja que una de un nivel inferior y que, además, la actividad está formada por actividades de un nivel inferior al suyo, sin necesidad de que sea el inmediatamente inferior.

Una vez definido el modelo jerárquico a seguir, se debe decidir si empezar a definir las actividades de manera ascendente (*bottom-up)* o descendente (*top-down)*. A continuación se hace un pequeño inciso para definir brevemente las metodologías anteriores [19]:

- Una metodología *bottom-up* se caracteriza por diseñar con detalle las partes individuales más pequeñas, en este caso los "eventos atómicos", formando seguidamente las partes más grandes, que a su vez forman otras más grandes, hasta formar el sistema completo.

- Una metodología *top-down* se caracteriza por formular primeramente el sistema completo, sin especificar detalle alguno. Posteriormente, se refina cada parte con mayor detalle hasta llegar a las partes más pequeñas. En este proyecto, se basaría en definir primeramente las maniobras a nivel general y, posteriormente, ir generalizando hasta obtener los eventos atómicos.

Tras haber analizado ambas soluciones, se concluye que la metodología *bottom-up* se adapta mejor al proyecto, ya que los eventos atómicos se pueden inferir directamente de las variables que se pueden extraer del simulador y de la cámara, por lo que su definición es casi inmediata. En nuestro modelo, un evento atómico es la actividad más simple, con una duración pequeña. Por ejemplo, se puede definir como "evento atómico" la actividad de "acelerar", la cual se infiere directamente del pedal del acelerador (variable "throotle").

Una vez se definan los eventos atómicos, estos se pueden combinar para formar "acciones", como por ejemplo la acción de "Meter primera marcha", compuesta de dos eventos atómicos: "pisar embrague" y "cambiar marcha superior". Seguidamente se definirán las "tareas" y, por último, las "maniobras".

3.2.1. Eventos atómicos

Tal y como se ha comentado anteriormente, se sigue una metodología *bottom-up*, definiendo primeramente las actividades situadas en la parte más inferior de la pirámide (ver Figura 19), en este caso, los eventos atómicos.

En el modelo propuesto, un evento atómico es una actividad simple (no está formada por otras actividades) y de corta duración (prácticamente inmediata). Tras haber estudiado las variables que se pueden extraer del simulador y de la cámara en el apartado 3.1. Viabilidad de datos, se definen los siguientes eventos atómicos, distinguiendo entre los eventos atómicos

inferidos de la interacción del conductor con el vehículo y los eventos atómicos inferidos de la posición de la cara y de los ojos del conductor.

EVENTOS ATÓMICOS CONDUCTOR	
EVENTO	**DESCRIPCIÓN**
pushGasPedal	Conductor pisa acelerador
releaseGasPedal	Conductor suelta acelerador
pushBreakPedal	Conductor pisa freno
releaseBreakPedal	Conductor suelta freno
pushClutchPedal	Conductor pisa embrague
changingGearUp	Conductor cambia marcha superior
changingGearDown	Conductor cambia marcha inferior
changingGearReverse	Conductor cambia marcha atrás
movingSW2Right	Conductor mueve volante a la derecha
movingSW2Left	Conductor mueve volante a la izquierda

Tabla 4: Eventos atómicos conductor

En la Tabla 4 se definen los eventos atómicos que realiza un conductor. Sin embargo, dentro de estos eventos atómicos no se incluyen los estados en los que se puede encontrar un vehículo. Que el conductor pise el acelerador (*pushGasPedal)* no implica que el coche acelere, ya que puede que esté en cuesta arriba o en una marcha demasiado larga. Por ello, es necesario definir los estados de un vehículo asociados a los eventos atómicos anteriores. Estos estados atómicos pueden denominarse "eventos atómicos vehículo", los cuales son, en definitiva, los estados del vehículo.

EVENTOS ATÓMICOS VEHÍCULO (ESTADOS VEHÍCULO)	
ESTADO	**DESCRIPCIÓN**
carThrootle	Coche acelera (velocidad en t > t-1)
carBraking	Coche frena (velocidad en t < t-1)
increasingRPM	Aumentan RPM (rpm en t > t-1)
decreasingRPM	Disminuyen RPM (rpm en t < t-1)

Tabla 5: Eventos atómicos vehículo

Para los eventos atómicos inferidos de la cámara, se definen las siguientes posiciones donde el conductor puede estar mirando:

- lookRightMirror: Mirando el retrovisor derecho.
- lookLeftMirror: Mirando el retrovisor izquierdo.
- lookAhead: Mirando hacia la carretera.
- lookRearMirror: Mirando retrovisor central.

Establecer a donde está mirando el conductor es una tarea sumamente compleja, ya que se cuenta con dos variables que proporcionan información: la posición de la cara y la posición de los ojos. Es necesario establecer una serie de reglas que indiquen, de manera muy

aproximada, hacia donde está mirando el conductor, utilizando la información de la cara y los ojos. Para ello, se decide utilizar conceptos de *lógica borrosa*, con cuantificadores como "fuertemente" o "débilmente". Por ejemplo, si la posición de la cara del conductor es "front-right" y la de los ojos es "right", entonces se asume que el conductor está claramente mirando al retrovisor derecho, por lo que se asume que el evento atómico es "lookRightMirrorStrongly" (mirando fuertemente al retrovisor derecho).

Los cuantificadores propuestos son "*strongly*", "very strongly", "weakly" y "very weakly". De esta manera, se tienen 16 posiciones difusas diferentes (4 posiciones * 4 cuantificadores). El razonamiento propuesto es el siguiente:

RAZONAMIENTO DIFUSO SOBRE POSICIONES DE CARA Y OJOS			
HEAD	**EYES**	**EVENTO ATÓMICO**	**COMENTARIOS**
FRONTHEAD-UP	RIGHT	*lookRearMirrorStrongly*	
FRONTHEAD-UP	CENTER	*lookRearMirrorWeakly*	
FRONTHEAD-UP	LEFT	*lookAheadWeakly*	
FRONTHEAD-UP	NoDetected	*lookRearMirrorWeakly*	Mirando carretera/retrovisor
FRONTHEAD-DOWN	Todas las posiciones	*lookAheadVeryWeakly*	Mirando al suelo/carretera
FRONTHEAD	CENTER	*lookAheadStrongly*	
FRONTHEAD	RIGHT/LEFT	*lookAheadWeakly*	También puede ser lookAheadWeakly
FRONT-LEFT	LEFT	*lookLeftMirrorStrongly*	
FRONT-LEFT	CENTER	*lookLeftMirrorWeakly*	
FRONT-LEFT	RIGHT/NO DETECTED	*lookLeftMirrorVeryWeakly*	También puede ser lookAheadWeakly
UP-LEFT	LEFT	*lookLeftMirrorWeakly*	
UP-LEFT	CENTER	*lookLeftMirrorVeryWeakly*	Tambien puede ser lookAheadWeakly
UP-LEFT	RIGHT/NO DETECTED	*lookAheadWeakly*	
DOWN-LEFT	Todas las posiciones	*lookLeftMirrorVeryWeakly*	
FRONT-RIGHT	RIGHT	*lookRightMirrorStrongly*	
FRONT-RIGHT	CENTER	*lookRightMirrorWeakly*	
FRONT-RIGHT	LEFT/NO-DETECTED	*lookRightMirrorVeryWeakly*	
UP-RIGHT	RIGHT	*lookRightMirrorWeakly*	
UP-RIGHT	CENTER	*lookRearMirrorStrongly*	
UP-RIGHT	LEFT/NO-DETECTED	*lookAheadWeakly*	
DOWN-RIGHT	Todas las posiciones	*lookRightMirrorVeryWeakly*	

Tabla 6: Eventos atómicos cámara

3.2.2. Acciones

El siguiente nivel jerárquico, dentro del modelo propuesto en la Figura 19, corresponde a las acciones. Se entiende por "Acción" una actividad compuesta por uno o varios eventos atómicos. La mayor parte de las acciones se definen en este modelo para dotar de mayor semántica al evento atómico, por lo que en la mayoría de los casos la cardinalidad es 1 a 1 entre acciones y eventos atómicos. Además, es posible que haya acciones compuestas de los mismos eventos atómicos.

A continuación se describen todas las acciones propuestas:

ACCIÓN	EVENTOS ATÓMICOS	COMENTARIOS
Subir marcha	releaseGasPedal clutched changingGearUp	1. Dejar de acelerar 2. Embragar 3. Cambiar marcha +
Reducir marcha	releaseGasPedal clutched changingGearDown	1. Dejar de acelerar 2. Embragar 3. Cambiar marcha -
Pisar acelerador	pushGashPedal	
Dejar de acelerar	releaseGasPedal	
Pisar freno	pushBrakePedal	
Dejar de frenar	releaseBrakePedal	
Mirar espejo interior	lookRearMirror	
Mirar espejo derecho	lookRightMirror	
Mirar espejo izquierdo	lookLeftMirror	
Girar volante derecha	movingSW2Right	
Girar volante izquierda	movingSW2Left	
Observar vehículo delantero	lookAhead	
Observar peatones	lookAhead	
Observar semáforo	lookAhead	

Tabla 7: Acciones

Como se observa, la clase "Acción" se introduce en el modelo con el objetivo de dotar de semántica a dicho modelo, ya que la mayoría de acciones se corresponden con un único evento atómico y, además, hay varias acciones que se componen del mismo evento atómico. Por ejemplo, observar el vehículo delantero y observar los peatones se realizan de la misma manera: mirando al frente de la carretera (*lookAhead)*.

3.2.3. Tareas

El siguiente nivel jerárquico del modelo propuesto se corresponde con las tareas. Una tarea, tal y como se ha comentado anteriormente, está formada por actividades de los niveles inferiores a una tarea, es decir, está compuesta de eventos atómicos y/o acciones. Estas tareas son más complejas que los niveles inferiores y requieren de un tiempo mayor de realización.

Estas actividades son ya lo sumamente complejas como para que se realicen sobre el simulador, generando los logs correspondientes para el posterior etiquetado de los mismos. Para realizar las tareas sobre el simulador, se deben generar unos escenarios con los

elementos necesarios para su realización. Por ejemplo, si se desea realizar una actividad que implica adelantar a un vehículo, se debe generar un escenario en el que el vehículo se encuentre detrás de otro vehículo y que este último se encuentre a una distancia tal que pueda ser adelantado.

Dado que generar un escenario de cero es una tarea sumamente compleja, se parte de la gran cantidad de escenarios que vienen definidos en el software del simulador del vehículo. Se estudia que escenario se adapta mejor a cada tarea y mediante el lenguaje de script del simulador (lenguaje generación de escenario), se introducen o se eliminan los elementos deseados.

Además de esto, se introduce el concepto de estado dentro de las tareas. Las tareas, como se ha comentado anteriormente, requieren de un tiempo de realización. Durante ese intervalo de tiempo, el vehículo (o los elementos del escenario) pasa por diferentes estados (*ver* Figura **20***).*

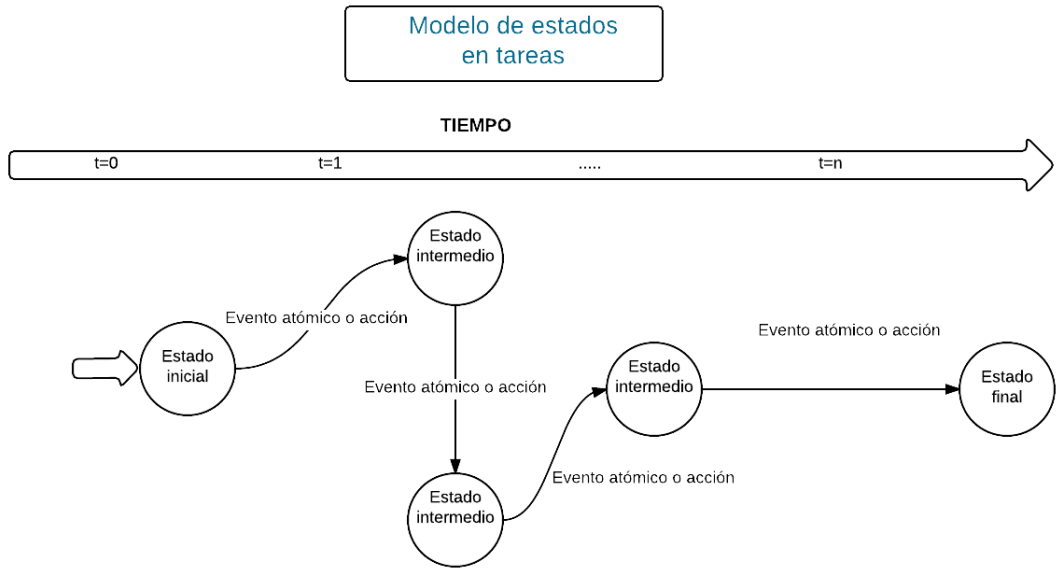

Figura 20**: Modelo de estados**

Donde los estados son:

- **Estado inicial**: es el estado en el que se encuentra el vehículo u otros elementos del entorno (escenario) antes de comenzar la tarea.
- **Estados intermedios**: conjunto de estados por los que pasa el vehículo u otros elementos del escenario.
- **Estado final**: es el estado en el que se encuentra el vehículo u otros elementos del escenario al finalizar la tarea.

Al realizar alguna acción o evento atómico, la tarea transita de un estado a otro, como si de una *máquina de estados* se tratase.

En este caso, la cardinalidad suele ser uno a muchos (1-*), es decir, a cada tarea le corresponden varias acciones y/o eventos atómicos.

Dado que las tareas tienen ya un nivel de complejidad alto, se definen brevemente de manera individual cada una de ellas, indicando y justificando qué variables se desean extraer del simulador para dicha tarea.

3.2.3.1. Arrancar y salir

Esta tarea se define para la situación en la que el coche se encuentra detenido y el conductor decide poner en marcha el vehículo y avanzar unos metros. A continuación, se muestra la composición de dicha tarea.

TAREA

TAREA	ACCION	EVENTO ATÓMICO
Arrancar y salir	Subir marcha	ReleaseGasPedal
		clutched
		changingGearUp
	Pisar acelerador	PushGasPedal

Tabla 8: Tarea arrancar y salir

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
SPEED=0 (DETENIDO) GEAR=0	increasingRevolutions carThrottle GEAR=1	SPEED >0 (EN MOVIMIENTO)

Tabla 9: Estados tarea arrancar y salir

Para esta tarea, inicialmente (*ver* Figura 21*)* el coche debe de estar detenido, por lo que su velocidad debe ser igual a 0, además de estar en punto muerto (gear=0).

Figura 21: Estado inicial arrancar-salir (en el simulador)

Seguidamente, se sube de marcha y se pisa el acelerador, incrementando las revoluciones del vehículo y acelerando. Finalmente (*ver* Figura 22*)*, el vehículo adquiere velocidad y se mueve hacia delante.

Figura 22: Estado final arrancar-salir

3.2.3.2. Detener coche

Esta tarea se define para la situación en la que un vehículo se encuentra en movimiento y el conductor decide detenerlo, frenando hasta que la velocidad del vehículo sea igual a 0. A continuación se muestra la composición de dicha tarea:

TAREA

TAREA	ACCION	EVENTO ATÓMICO
Detener coche	Dejar de acelerar	ReleaseGasPedal
	Pisar freno	PushBreakPedal
	Reducir marcha	ReleaseGasPedal
		clutched
		changingGearDown

Tabla 10: Tarea detenerse

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
SPEED >0 (EN MOVIMIENTO)	carBraking decreasingRevolutions	SPEED=0 (DETENIDO)

Tabla 11: Estados detenerse

Para esta tarea, inicialmente (*ver* Figura 23*)* el vehículo debe de estar en movimiento, por lo que su velocidad debe de ser mayor a 0.

Figura 23: Estado inicial tarea detenerse

Seguidamente, se debe dejar de acelerar y pisar el pedal del freno, por lo que el coche frenará y se reducirán las revoluciones. Finalmente (*ver* Figura 24*)*, se debe de reducir de marcha mientras que se mantiene el freno, hasta que el vehículo se detenga.

Figura 24: Estado final tarea detener coche

3.2.3.3. Aumentar/reducir velocidad

Estas tareas indican las situaciones en las que se aumenta o se reduce la velocidad. Estas tareas se consideran especiales, ya que forman parte de actividades de nivel superior (maniobras) y no se consideran propiamente una tarea como tal a realizar. Sin embargo, se ha decidido definirlas dado que están compuestas por acciones y eventos atómicos. Por estos motivos, no se van a realizar sobre el simulador, sino que se realizarán intrínsecamente dentro de las maniobras que se definen en apartados posteriores.

A continuación se muestra su composición:

TAREAS

TAREA	ACCION	EVENTO ATÓMICO
Aumentar velocidad	Pisar acelerador	PushGasPedal
	Aumentar marcha	ReleaseGasPedal
		clutched
		changingGearUp
Reducir velocidad	Dejar de acelerar	ReleaseGasPedal
	Pisar freno	PushBrakePedal
	Reducir marcha	ReleaseGasPedal
		clutched
		changingGearDown

Tabla 12: Tareas aumentar/reducir velocidad

Para la tarea de aumentar velocidad, se pisa el acelerador hasta que el coche adquiere unas revoluciones altas y posteriormente se cambia de marcha. Para dicha acción, se suelta el acelerador, se embraga y se aumenta la marcha mediante la palanca.

Para la tarea de reducir velocidad, se deja de acelerar, se pisa el freno y se reduce de marcha para evitar que el coche “se cale”.

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL	COMENTARIOS
speed en t1	*increasingRevolutions* *carThrottle*	speed en t2	speed en t2 > speed en t1
speed en t1	*decreasingRevolutions* *carBraking*	speed en t2	speed en t2 < speed en t1

Tabla 13: Estados tarea aumentar/reducir velocidad

3.2.3.4. Marcha atrás

Esta tarea se define para la situación en la que el conductor decide ir marcha atrás para, por ejemplo, realizar un aparcamiento o salir de un parking. Para que un vehículo vaya marcha atrás, es necesario que esté detenido para poder introducir la marcha "7" (marcha atrás del simulador) y, posteriormente, acelerar. Esta aceleración no desplaza el vehículo hacia delante sino hacia atrás. A continuación se muestra la composición de dicha tarea:

TAREA

TAREA	ACCION	EVENTO ATÓMICO
Marcha atrás	Mirar espejo interior	LookRearMirror
	Reducir marcha	pushClutchPedal
		changingGearDown
	Pisar acelerador	PushGasPedal

Tabla 14: Tarea marcha atrás

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
SPEED=0 (DETENIDO)	Gear= R (marcha atrás) *increasingRevolutions* *carThrottle*	SPEED >0 (EN MOVIMIENTO)

Tabla 15: Estados marcha atrás

Para esta tarea, el vehículo debe de estar inicialmente detenido (velocidad igual a 0), ya que es una condición indispensable para poder introducir la marcha atrás (*ver* Figura 25*)*.

Figura 25: Estado inicial tarea marcha atrás

Seguidamente, se introduce la marcha atrás embragando y metiendo la marcha 7 (marcha atrás en el simulador, R en la interfaz). Finalmente (*ver* Figura 26*)*, se acelera, incrementando su velocidad y revoluciones.

Figura 26: Estado final tarea marcha atrás

3.2.3.5. Cambiar carril derecho/ cambiar carril izquierdo

Estas tareas se explican de manera conjunta ya que lo único que cambia es el carril al que el conductor quiere desplazarse. Para cambiarse de carril es necesario observar por el retrovisor correspondiente si es seguro cambiarse de carril, girar el volante hacia el carril deseado y, por último, enderezar el volante girándolo hacia el lado contrario. A continuación se muestra la composición de dichas tareas.

TAREAS

TAREA	ACCIÓN	EVENTO ATÓMICO
Cambiar carril derecha	Mirar espejo derecha	lookRightMirror
	Girar volante derecha	movingSW2Right
	Girar volante izquierda	movingSW2Left
Cambiar carril izquierda	Mirar espejo izquierda	lookLeftMirror
	Girar volante izquierda	movingSW2Left
	Girar volante derecha	movingSW2Right

Tabla 16: Tareas cambiar carril

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
LANE= left		LANE=right
LANE=right		LANE= left

Tabla 17: Estados tarea cambiar de carril

Para estas tareas, el vehículo debe de estar inicialmente (*ver* Figura 27*)* en uno de los carriles (en caso de la tarea de cambiar al carril derecho, el vehículo debe de estar en el carril izquierdo, y en caso de querer cambiar al carril izquierdo, el vehículo debe de estar en el derecho).

Figura 27: Estado inicial tarea cambiar carril derecho

Seguidamente, el conductor debe mirar al retrovisor correspondiente y girar el volante al carril deseado. Por último, el conductor debe de enderezar el volante, girándolo al lado contrario de donde lo había girado previamente. Finalmente (*ver* Figura 28*)*, el vehículo se encontrará en el carril deseado.

Figura 28: Estado final cambio carril derecho

3.2.4. Maniobras

El último escalafón dentro del modelo propuesto corresponde a las maniobras. Una maniobra es una actividad formada por tareas, acciones y eventos atómicos. Se diferencian de las tareas en que toda maniobra lleva asociada al menos una tarea, por lo que la maniobra se encuentra por encima en el modelo jerárquico. Además, se trata de la actividad más compleja y duradera de las cuatro actividades del modelo propuesto.

Al igual que sucede con las tareas, las maniobras se realizan sobre el simulador para generar los logs que posteriormente se utilizarán como conjuntos de datos de entrenamiento. Con esto se pretende generar un modelo que detecte automáticamente estas maniobras.

El procedimiento a seguir es exactamente igual que el propuesto para las tareas: se definen unos escenarios ideales para las maniobras ideales, estudiando y seleccionando qué variables son interesantes de extraer en cada tiempo definido, y el investigador realiza cada maniobra en el simulador. Además, igual que sucede con las tareas, el vehículo va pasando por diferentes estados mientras se realiza la maniobra, por lo que cada maniobra sigue una *máquina de estados* (verFigura **20**).

A continuación se definen de manera individual cada una de las maniobras.

3.2.4.1. Adelantar

Esta maniobra consiste en adelantar a un vehículo de la manera natural, es decir, por el carril de la izquierda (se asume que la conducción se realiza por el carril de la derecha). Adelantar a un vehículo se compone de varias tareas, acciones y eventos atómicos. De manera general, para adelantar a un vehículo que circula delante de nosotros es necesario observar la distancia del vehículo respecto a nuestro vehículo, cambiarse al carril de la izquierda, acelerar para sobrepasarle y, por último, volver al carril normal de conducción, es decir, al carril derecho.

A continuación se muestra la composición de dicha maniobra, así como los diferentes estados por los que pasa el vehículo.

MANIOBRA

MANIOBRA	TAREA	ACCIÓN	EVENTO ATÓMICO
Adelantar		Observar vehículo delante	lookAhead
	Cambiar carril izquierda	Mirar espejo izquierda	lookLeftMirror
		Girar volante izquierda	movingSW2Left
		Girar volante derecha	movingSW2Right
	Aumentar velocidad	Pisar acelerador	PushGasPedal
		Aumentar marcha	ReleaseGasPedal
			clutched
			changingGearUp
	Cambiar carril derecho	Mirar espejo derecha	lookRightMirror
		Girar volante derecha	movingSW2Right
		Girar volante izquierda	movingSW2Left

Tabla 18: Maniobra adelantar

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
leadingVehicle.presence=YES	carThrottle	behindVehicle=leadingVehicle
leadingVehicle.distance<50	increasingRevolutions	"Adelantado"
speedRelated2driver < speed		

Tabla 19: Estados adelantar

Para esta maniobra, inicialmente (*ver* Figura **29***)* el conductor debe de observar el vehículo de delante (*lookAhead)* para saber si es susceptible de ser adelantado o no. El estado inicial, por lo tanto, es que haya un vehículo delante, a una distancia cercana (menos de 50 metros) y que su velocidad sea inferior a la de nuestro vehículo.

Figura 29: **Estado inicial maniobra** *Adelantar*

Si se dan esos condicionantes, entonces el conductor debe de cambiarse al carril de la izquierda (*cambiar carril izquierdo)* y *aumentar la velocidad*, lo que conlleva que el vehículo acelere y aumenten las revoluciones (*ver* Figura 30*)*.

Figura 30: Estado intermedio maniobra adelantar

Una vez adelantado, el conductor comprueba si ha adelantado al vehículo o no. Para ello, debe de mirar al retrovisor derecho (*lookRightMirror).* Por último, el conductor gira al carril habitual de conducción (*cambiar carril derecho*) (*ver* Figura 31*)*.

Figura 31: Estado final maniobra adelantar

3.2.4.2. Ceder paso de peatones

Esta maniobra se basa en ceder el paso a peatones en un paso de peatones, comúnmente llamado "paso de cebra". Para esta maniobra, se asume que existen peatones cruzando el paso de cebra y que, por lo tanto, el vehículo tendrá que detenerse por completo. Por lo tanto, el conductor deberá observar el paso de peatones, ir reduciendo velocidad y, por último, detenerse para ceder el paso de los peatones.

A continuación se describe la composición de la maniobra.

MANIOBRA

MANIOBRA	TAREA	ACCIÓN	EVENTO ATÓMICO
Ceder paso de peatones		Observar vehículo trasero	lookRearMirror
	Reducir Velocidad	Dejar de acelerar	ReleaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	ReleaseGasPedal
			clutched
			changingGearDown
		Observar paso peatones	lookAhead
	Detener coche	Dejar de acelerar	ReleaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	pushClutchPedal
			changingGearDown

Tabla 20: Maniobra ceder paso peatones

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
pedestrian.presence=YES **pedestrian.crossing=YES**	braking decreasingRevolutions	SPEED=0 (DETENIDO)

Tabla 21: Estados paso de peatones

Inicialmente, debe de haber peatones cruzando un paso de cebra y el vehículo del conductor aproximándose a él (*ver* Figura 32*)*. El conductor, antes de reducir la velocidad, debe de observar, por el retrovisor central si hay algún vehículo cercano detrás suyo, por seguridad.

Figura 32: Estado inicial paso de peatones

Seguidamente, el conductor debe reducir la velocidad (*reducir velocidad)*, lo que implica la bajada de revoluciones del vehículo y el freno del mismo. Por último, el conductor debe de observar si hay peatones cruzando (*ver* Figura **33***)*. En ese caso, el conductor debe detenerse (*detener coche)* y ceder el paso. Si no hubiera peatones, el conductor debe continuar la marcha.

Figura 33: **Estado final paso de peatones**

3.2.4.3. Mantener distancia seguridad

Esta maniobra consiste en mantener la distancia de seguridad entre el vehículo del conductor y un vehículo que circule delante del mismo. La distancia mínima de seguridad es aquella que permite a un conductor reaccionar cuando hay que evitar una colisión contra un vehículo que circula delante de él. En este caso, se ha establecido una distancia mínima de 50 metros, por lo que si un vehículo que circula delante del conductor se encuentra a menos de esa distancia, entonces el conductor realizará esta maniobra.

A continuación se describe la composición de la maniobra:

MANIOBRA

MANIOBRA	TAREA	ACCIÓN	EVENTO ATÓMICO
Mantener distancia seguridad		Observar vehículo delante	lookAhead
	Reducir velocidad	Dejar de acelerar	ReleaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	ReleaseGasPedal
			pushClutchPedal
			changingGearDown

Tabla 22: **Maniobra distancia seguridad**

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
leadingVehicle.presence=YES **leadingVehicle.distance<50** **speedRelated2driver < speed**	braking decreasingRevolutions	speedRelated2Driver>=speed distance>50

Tabla 23: Estados mantener distancia seguridad

Inicialmente, debe de haber un vehículo delante del vehículo del conductor (denominado *leadingVehicle)*. Además, dicho vehículo debe de encontrarse a una distancia menor de 50 metros y su velocidad debe de ser inferior a la del vehículo del conductor (*ver* Figura 34*)*. En ese caso, el vehículo del conductor se aproxima al vehículo y, en caso de no reducir su velocidad, colisionará con el vehículo.

Figura 34: Estado inicial mantener distancia seguridad

Una vez el vehículo se aproxima al conductor, el conductor debe de reducir la velocidad (*reducir velocidad)*, lo que implica una bajada de las revoluciones y el freno del vehículo. Finalmente, el vehículo del conductor debe de quedarse a una distancia mayor de 50 metros y su velocidad debe de mantenerse constante, siempre y cuando sea menor o igual a la del vehículo que le sucede. Como se observa en (*ver* Figura 35*)*, en este ejemplo, la velocidad para mantener la distancia es de unos 35 km/h.

Figura 35: Estado final mantener distancia

3.2.4.4. Detenerse ante semáforo en rojo

Esta maniobra se basa en detenerse ante un semáforo que se encuentra en rojo y, por lo tanto, está prohibido sobrepasarlo hasta que no pase a verde. Un conductor, habitualmente, observa el semáforo a una distancia prudencial y, si el semáforo se encuentra en rojo, detiene el vehículo. A continuación se muestra la composición de la maniobra.

MANIOBRA

MANIOBRA	TAREA	ACCION	EVENTO ATÓMICO
Detenerse ante semáforo		Observar semáforo	lookAhead
	Detener coche	Dejar de acelerar	releaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	releaseGasPedal
			pushClutchPedal
			changingGearDown

Tabla 24: Maniobra detenerse semáforo

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
speed > 0 (MOVIMIENTO)	braking	speed=0 (DETENIDO)
TrafficLigh.distance >0	decreasingRevolutions	
TrafficLigh.color = RED	clutched	
TrafficLigh.distance <50		

Tabla 25: Estados detenerse semáforo

Primeramente, el vehículo debe de estar en movimiento (velocidad mayor que 0), debe de haber un semáforo cercano (a menos de 50 metros) y debe de estar en rojo (ver Figura **36***)*. Ante esa situación, el conductor debe de observar el color del semáforo para

decidir cómo actuar: en caso de estar en verde, el conductor debe de continuar y, en caso de estar en rojo, el conductor debe de detenerse.

Figura 36: Estado inicial detenerse semáforo

En el ejemplo (*ver* Figura 37*)*, el semáforo está en rojo, por lo que el conductor debe de detener el coche, reduciéndose las revoluciones del vehículo, reduciendo la marcha del vehículo y, por último, deteniéndose.

Figura 37: Estado final detenerse semáforo

3.2.4.5. Aparcar en línea a izquierdas/derechas

Estas dos maniobras se describen de manera conjunta puesto que son la misma maniobra, a excepción del lugar de aparcamiento: en una maniobra se aparca “a izquierdas”, es decir, en un hueco situado a la izquierda del vehículo, y en otra se aparca “a derechas”, es decir, en un hueco situado a la derecha del vehículo.

Se trata de una maniobra sumamente compleja, ya que hay que realizar una gran cantidad de tareas, acciones y eventos atómicos. Además, muchos eventos atómicos se realizan de manera concurrente, como puede ser mirar a los retrovisores mientras se gira el volante. A continuación se muestran la composición de estas dos maniobras:

MANIOBRA

MANIOBRA	TAREA	ACCION	EVENTO ATÓMICO
Aparcar a derechas en línea		Girar volante derecha	movingSW2Right
	Marcha atrás	Mirar espejo interior	LookRearMirror
		Reducir marcha	pushClutchPedal
			changingGearDown
		Acelerar	PushGasPedal
		Mirar espejo derecha	lookRightMirror
	Girar izquierda	Girar volante izquierda	movingSW2Left
		Mirar espejo izquierda	lookRightMirror
	Detener coche	Dejar de acelerar	
		Pisar freno	PushBreakPedal
		Reducir marcha	ReleaseGasPedal
			pushClutchPedal
			changingGearDown

Tabla 26: Maniobra aparcar línea derecha

MANIOBRA	TAREA	ACCION	EVENTO ATÓMICO
Aparcar a izquierdas en línea		Girar volante izquierda	movingSW2Left
	Marcha atrás	Mirar espejo interior	LookRearMirror
		Reducir marcha	ReleaseGasPedal
			pushClutchPedal
			changingGearDown
		Acelerar	PushGasPedal
		Mirar espejo izquierda	lookLeftMirror
		Girar volante derecha	movingSW2Right
		Mirar espejo derecha	lookRightMirror
	Detener coche	Dejar de acelerar	ReleaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	ReleaseGasPedal
			pushClutchPedal
			changingGearDown

Tabla 27: Maniobra aparcar línea izquierda

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
speed=0 (DETENIDO) huecoAparcar=TRUE coche paralelo al hueco hueco > 5 metros	gear = R (marcha atrás) throttle increasingRevolutions clutched decreasingRevolutions	speed=0 (APARCADO)

Tabla 28: Estados aparcar línea

La siguiente figura (ver Figura 38) muestra de manera gráfica cómo se realiza un aparcamiento en línea:

Figura 38: aparcamiento línea [3]

Inicialmente, el vehículo se debe de encontrar detenido en paralelo al hueco donde se desea aparcar el coche. Generalmente este hueco será el formado entre dos vehículos previamente estacionados. Este hueco debe de ser al menos de 5 metros para que cualquier vehículo normal pueda estacionarse (*ver* Figura 39*)*.

Figura 39: Estado inicial aparcamiento linea

Una vez el vehículo se encuentra detenido en paralelo al hueco, se gira el volante a la izquierda y se circula marcha atrás, todo ello mientras por el espejo retrovisor central y el retrovisor izquierda se observa que no se ponga en peligro la seguridad (*ver* Figura 40*)*.[1]

Figura 40: Estado intermedio aparcamiento línea

Una vez la parte trasera del coche se encuentra dentro del hueco, se gira el volante para la derecha con el objetivo de alinea el vehículo con el vehículo delantero. Mientras tanto, por el retrovisor derecho se observa que el coche quede alineado.

Por último, el coche se detiene (*detener coche) (ver* Figura 41*)*

Figura 41: Estado final aparcamiento línea

3.2.4.6. Aparcar batería frente derecha/izquierda

Estas dos maniobras se describen de manera conjunta puesto que son la misma maniobra, a excepción del lugar de aparcamiento: en una maniobra se aparca “a izquierdas”, es decir, en un hueco situado a la izquierda del vehículo, y en otra se aparca “a derechas”, es decir, en un hueco situado a la derecha del vehículo.

Estas maniobras son menos complejas que aparcar en línea, ya que no es necesario circular marcha atrás. Aparcar en batería es tan sencillo como reducir la velocidad y girar hacia

[1] Las rayas rojas, amarillas y verdes contenidas en la figura se deben a que el simulador las muestra cuando se da marcha atrás.

el hueco deseado. En este caso, se va a seguir el ejemplo de aparcar de frente en batería a la derecha.

MANIOBRA

MANIOBRA	TAREA	ACCION	EVENTO ATÓMICO
Aparcar batería frente derecha	Reducir velocidad	Dejar de acelerar	releaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	releaseGasPedal
			pushClutchPedal
			changingGearDown
		Girar volante derecha	movingSW2Right
	Detener coche	Dejar de acelerar	ReleaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	ReleaseGasPedal
			pushClutchPedal
			changingGearDown
Aparcar batería frente izquierda	Reducir velocidad	Dejar de acelerar	releaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	releaseGasPedal
			pushClutchPedal
			changingGearDown
		Girar volante izquierda	movingSW2Left
	Detener coche	Dejar de acelerar	ReleaseGasPedal
		Pisar freno	PushBreakPedal
		Reducir marcha	ReleaseGasPedal
			pushClutchPedal
			changingGearDown

Tabla 29: Maniobras aparcar batería

ESTADOS

ESTADO INICIAL	ESTADOS INTERMEDIO	ESTADO FINAL
Speed > 0 (movimiento) *huecoAparcar=TRUE*	*carBraking* *gear=1 / 2* *decreasingRPM*	*Speed =0 (detenido)*

Tabla 30: Estados aparcar batería

Inicialmente, el vehículo debe de estar en movimiento y debe de haber un hueco para aparcar en el lado deseado (*ver* Figura 42*)*. En el caso de que haya un hueco, el conductor debe

de *reducir la velocidad* mientras se acerca al hueco. Para el ejemplo, se asume que el hueco está entre dos coches.

Figura 42: Estado inicial aparcar batería

Seguidamente, el conductor debe de girar el volante hacia el lado donde desea aparcar, una vez se haya superado al primer vehículo. La marcha debe de ser baja, primera o segunda, ya que la velocidad es muy baja también (*ver* Figura 43*)*. Esto se realiza para que el coche no "se cale".

Figura 43: Estado intermedio aparcar batería

Por último, el conductor debe de *detener* el vehículo, por lo que en el estado final (*ver*Figura 45 Figura 44*)* el vehículo se encuentra estacionado entre los dos vehículos.

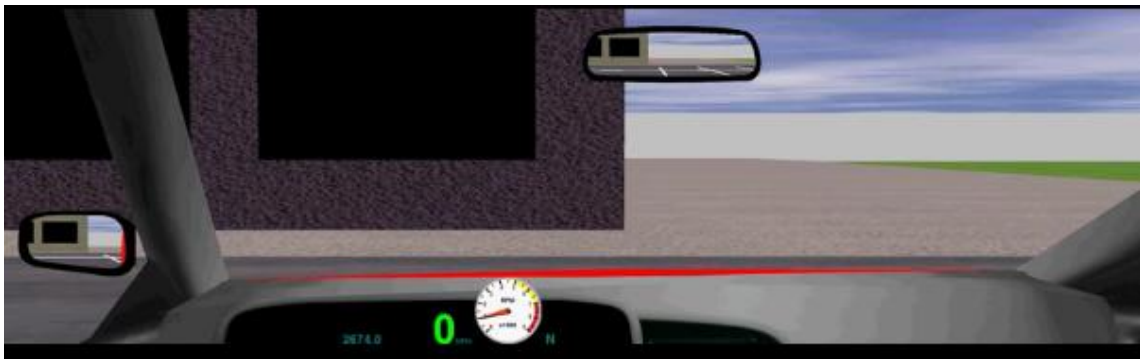

Figura 44: Estado final aparcar batería

Capítulo 4: Análisis y Diseño del Sistema

4. Análisis y diseño del sistema

En este apartado de la memoria se va a analizar y describir de manera explícita y detallada todo el sistema a desarrollar. Primeramente, se detallan los estándares, normas o recomendaciones que debe de seguir el sistema, así como el entorno operacional del mismo. Seguidamente, se especifican los diferentes tipos de requisitos que debe cumplimentar el sistema, así como los distintos casos de uso del mismo. Por último, se describe el diseño del sistema.

4.1. Normas y restricciones del proyecto

En este apartado se especifican las normas y restricciones a cumplimentar por el sistema, así como el entorno operacional.

Las **normas** que se deben cumplir son las siguientes:

- Dado que el sistema utiliza datos personales del usuario (se graba en vídeo la cara del usuario), será necesario cumplir con la Ley Orgánica 15/1999 de Protección de Datos de Carácter Oficial [20].

Las **restricciones** que se deben cumplir se dividen en restricciones de hardware y restricciones de software.

Las *restricciones de hardware* son las siguientes:

- La cámara a utilizar para grabar la cara del usuario debe ser una Kinect XBOX360, pudiéndose utilizar en un futuro la Kinect 2.0.
- La cámara de vídeo debe estar conectada a un ordenador distinto al ordenador donde está instalado el software del simulador.
- Se debe de contar con un simulador STISIM M100 Series.
- El simulador STISIM M100 Series debe de contar con pedales de aceleración, freno y embrague, volante, palanca de cambios y unos monitores que simulen la luna delantera de un vehículo.
- El ordenador donde se implanta el sistema debe de estar conectado al simulador mediante los cables correspondientes.
- El módulo del sistema y el módulo conectado a la cámara se conectan mediante un socket, utilizando la arquitectura cliente-servidor.

Las *restricciones de software* son las siguientes:

- El ordenador donde se conecta el simulador debe de contar con el software asociado (STISIM Drive 3.03.08 o superior).
- El ordenador conectado a la cámara debe de contar con Microsoft Visual Studio 2008 para poder ejecutar el programa de detección de cara y ojos (escrito en C++).
- JDK para poder ejecutar el programa escrito Java que trata los datos generados.
- Entorno de programación en Java.

4.2. Entorno operacional

En este apartado se define el entorno operacional del proyecto, es decir, se enumeran las herramientas de software y de hardware utilizadas durante el desarrollo del proyecto:

- Cámara Kinect XBOX360.
- Simulador STISIM M100 Series.
- Ordenador LG con procesador Inter Core I7-3440 3,4Ghz y 16GB de RAM.
- Pedales, volante y palanca de cambios de la marca Logitech.
- Librerías OpenCV para C++, que incluye numerosas funciones del ámbito de la visión por computador.
- Entorno de desarrollo de C++ Microsoft Visual Studio 2008.
- Entorno de desarrollo Java Eclipse Kepler 4.3.
- Paquete ofimático de Microsoft Office 2010 y 2013 (Word y Excel).
- Hypercam 2.29. Software dedicado a grabar la actividad de una pantalla de Windows en formato AVI.
- Lucidchart. Aplicación web que permite realizar diagramas.

4.3. Especificación de requisitos

En este apartado se especifican los diferentes tipos de requisitos que debe de cumplimentar el proyecto a desarrollar. Para cada uno de los requisitos se va a elaborar una tabla. Dicha tabla tendrá el siguiente formato:

RY-X		
PRIORIDAD: ☐Alta	☐ Media	☐ Baja
NECESIDAD: ☐ Esencial	☐ Deseable	☐ Opcional
CLARIDAD: ☐ Alta	☐ Media	☐ Baja
DESCRIPCIÓN:		

Tabla 31: Modelo requisitos

Dónde:

- **RY-X:** representa el identificador del requisito, siendo:
  - R: requisito.
  - Y: requisito funcional 'F' o requisito no funcional 'NF'.
  - X: número entero para el identificador.

- **IDENTIFICACIÓN**: cada requisito de usuario incluirá una identificación, para facilitar la trazabilidad.
- **NECESIDAD**: Representa la importancia del requisito para el desarrollo del proyecto. Existen tres alternativas diferentes: esencial, deseable y opcional. Un requisito esencial es imprescindible su cumplimentación para el desarrollo del proyecto, es decir, en caso de no cumplirse, el proyecto no podría ser llevado a cabo. Un requisito deseable es aquel que se desea cumplir pero que no se asegura su cumplimentación, ya que depende de diversos factores. Por último, un requisito opcional es aquel que puede o no ser cumplido, es decir, aquel que no es estrictamente necesario cumplimentar para el desarrollo del proyecto.
- **PRIORIDAD**: cada requisito de usuario incluirá una medida de la prioridad que posee, para que el desarrollador pueda decidir la planificación del desarrollo.
- **CLARIDAD**: un requisito de usuario es claro si tiene una, y sólo una, interpretación, por lo que tendrá claridad "Alta". En caso de que el requisito de lugar a pequeñas ambigüedades, la claridad será "Media". Por último, si el requisito es totalmente ambiguo, tendrá una claridad "Baja".

4.3.1. Requisitos funcionales

RF-001		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema deberá escribir en un fichero una estimación de la posición de la cara del conductor en cada instante de tiempo definido, dentro de la realización de una actividad ideal por parte del usuario.		

Tabla 32: RF-001

RF-002		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema deberá escribir en un fichero una estimación de la posición de los ojos del conductor en cada instante de tiempo definido, dentro de la realización de una actividad ideal por parte del usuario.		

Tabla 33: RF-002

RF-003		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: Las estimaciones de la cara pueden ser: fronthead, front-right, front-left, front-up, front-down, up-right, up-left, down-right y down-left.		

Tabla 34: RF-003

RF-004		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: Las estimaciones de los ojos pueden ser: center, right o left.		

Tabla 35: RF-004

RF-005		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema solo podrá definir una única posición para la cara. Es decir, no están permitidas posiciones concurrentes.		

Tabla 36: RF-005

RF-006		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema solo podrá definir una única posición para los ojos. Es decir, no están permitidas posiciones concurrentes.		

Tabla 37: RF-006

RF-007		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema, una vez finalizada la simulación de una actividad de conducción, deberá generar de manera automática un fichero de log.		

Tabla 38: RF-007

RF-008		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El fichero de log contendrá instancias que indiquen los valores de una serie de variables (acelerador, freno, ángulo de volante, etc.) que se puedan extraer del simulador.		

Tabla 39: RF-008

RF-009		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: Para cada par de ficheros de "log" y de "estimación de cara y ojos", el sistema deberá generar un fichero combinado de estos dos, denominado "conjunto de entrenamiento sin procesar".		

Tabla 40: RF-009

RF-010		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: Para la generación de cada conjunto de entrenamiento sin procesar, el sistema contará con un método que reciba como parámetro las rutas de los ficheros a juntar, así como la ruta del fichero de salida generado.		

Tabla 41: RF-010

RF-011		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: Cada instancia o fila de cada conjunto de entrenamiento sin procesar deberá contener los valores de las variables generadas para cada instante de tiempo, así como una posición para la cara del conductor y una posición para los ojos del conductor.		

Tabla 42: RF-011

RF-012		
PRIORIDAD:		
☐Alta	☒ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☐ Deseable	☒ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema contará con un método que permita normalizar los datos entre 0 y 1 que requiera el usuario.		

Tabla 43: RF-012

RF-013		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema contará con un método que permita almacenar en memoria de manera temporal los datos de un fichero (lectura de fichero).		

Tabla 44: RF-013

RF-014		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema contará con un método que permita razonar con los datos almacenados en memoria, reconociendo de manera automática en cada instante de tiempo el evento atómico (o eventos) que está realizando el conductor.		

Tabla 45: RF-014

RF-015		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: Los eventos atómicos pueden ser: carThrootle, carBraking, increasingRPM, decreasingRPM, pushGasPedal, releaseGasPedal, pushBreakPeda, releaseGasPedal, movingSW2Right, movingSW2Left, clutched, changingGearReverse, changingGearUp y changingGearDown.		

Tabla 46: RF-015

RF-016		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El sistema deberá contener una funcionalidad que permita razonar con las posiciones de cara y ojos del conductor, estableciendo hacia donde está mirando el usuario en otro evento atómico distinto.		

Tabla 47: RF-016

RF-017		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El sistema escribirá en un fichero de salida (conjunto de entrenamiento procesado) todas las variables almacenadas en memoria (las normalizadas y las que no), el evento atómico (o eventos) que está realizando el conductor y el evento que indique hacia donde está mirando el usuario, para cada instancia almacenada en memoria.		

Tabla 48: RF-017

RF-018		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El sistema escribirá una cabecera en cada conjunto de entrenamiento procesado. Esta cabecera se lee de otro un fichero de cabeceras.		

Tabla 49: RF-018

RF-019		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: Se deberá aplicar un *trie* sobre los eventos atómicos generados, a fin de estudiar las secuencias de eventos atómicos generados (modelo de tareas).		

Tabla 50: RF-019

4.3.2. Requisitos no funcionales

RNF-001		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El investigador debe definir de manera clara y concisa qué es una actividad, una maniobra, una tarea y una acción atómica.		

Tabla 51: RNF-001

RNF-002		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El investigador debe definir el modelo de actividades que realiza un conductor al volante en un formato "Excel" o similar.		

Tabla 52: RNF-002

RNF-003		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: Para cada actividad ideal definida, el investigador deberá definir un escenario ideal donde realizar dicha tarea.		

Tabla 53: RNF-003

RNF-004		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: Los escenarios ideales se definen en el lenguaje de script (*Scenario Definition Languaje)* propio del simulador STISIM.		

Tabla 54: RNF-004

RNF-005		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El investigador deberá realizar las actividades ideales con el simulador, realizándose éstas sobre los escenarios ideales definidos.		

Tabla 55: RNF-005

RNF-006		
PRIORIDAD:		
☐Alta	☐ Media	☒ Baja
NECESIDAD:		
☐ Esencial	☐ Deseable	☒ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: La realización de las actividades deberá ser grabada en vídeo por algún software de grabación de pantalla, a fin de documentar esta información.		

Tabla 56: RNF-006

RNF-007		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: La información que el sistema genere en los ficheros de log deberá ser indicada por el investigador en la definición del escenario.		

Tabla 57: RNF-007

RNF-008		
PRIORIDAD:		
☐Alta	☒ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El investigador deberá estudiar qué variables deben ser escritas en el fichero log para cada una de las actividades.		

Tabla 58: RNF-008

RNF-009		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: Los ficheros de log generados deberán estar en formato “csv” (ficheros separados por “;”).		

Tabla 59: RNF-009

RNF-010		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El investigador deberá realizar un pre-procesado de los ficheros de manera manual, eliminando información irrelevante que genera automáticamente el software del simulador.		

Tabla 60: RNF-010

RNF-011		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☒ Esencial	☐ Deseable	☐ Opcional
CLARIDAD:		
☒ Alta	☐ Media	☐ Baja
DESCRIPCIÓN: El sistema deberá escribir en un fichero la posición de cara y ojos del conductor cada 0.033 segundos.		

Tabla 61: RNF-011

RNF-012		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El sistema deberá escribir en un fichero el valor de las variables seleccionadas cada 0.033 segundos.		

Tabla 62: RNF-012

RNF-013		
PRIORIDAD:		
☒Alta	☐ Media	☐ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El sistema deberá generar el conjunto final de entrenamiento de cada actividad en un tiempo máximo de 2 segundos desde que el usuario deje de ejecutar la aplicación.		

Tabla 63: RNF-013

RNF-014		
PRIORIDAD:		
☐Alta	☐ Media	☒ Baja
NECESIDAD:		
☐ Esencial	☒ Deseable	☐ Opcional
CLARIDAD:		
☐ Alta	☒ Media	☐ Baja
DESCRIPCIÓN: El investigador deberá generar un manual de usuario donde explique cómo hacer funcionar el sistema.		

Tabla 64: RNF-014

4.4. Casos de uso

En este apartado se definen los casos de uso del sistema. Un caso de uso es una descripción de los pasos que debe de seguir un actor (personaje que realiza la acción) para la realización de una tarea concreta.

Los casos de uso se pueden definir tanto de manera tabular como de manera gráfica (en forma de diagrama de caso de uso).

4.4.1. Descripción tabular de casos de uso

A continuación se presentan los casos de uso de manera tabular. Las tablas usadas siguen el siguiente formato:

CU-X: NOMBRE
ACTOR:
OBJETIVO:
PRECONDICIONES:
POSTCONDICIONES:
ESCENARIO DE ÉXITO:

Tabla 65: Ejemplo tabla caso de uso

Dónde:

- **CU-X: NOMBRE:** representa el identificador del caso de uso, siendo:
  - CU: Caso de Uso.
  - X: número natural que enumera los casos de uso, siendo el primero el “001”.
  - NOMBRE: nombre con el que se identifica el caso de uso.

- **ACTOR**: tipo de usuario que realiza el caso de uso. En este caso, el investigador es el que realiza todas las tareas.
- **OBJETIVO**: descripción de la finalidad del caso de uso.
- **PRECONDICIONES**: condiciones que deben cumplir para poder realizar el caso de uso.
- **POSTCONDICIONES**: estado en que queda el sistema tras realizar el caso de uso.
- **ESCENARIO DE ÉXITO**: pasos para cumplir el objetivo

CU-001: Discretizar valores
ACTOR: Investigador
OBJETIVO: Discretizar las posiciones de cara y ojos devueltas por la aplicación desarrollada en el departamento de Automática.
PRECONDICIONES: • La aplicación retorna un valor numérico para la posición de la cara • La aplicación retorna un valor numérico para la posición de los ojos
POSTCONDICIONES: • Los valores quedan discretizados en posiciones absolutas para la cara (valor alfabético). • Los valores quedan discretizados en posiciones absolutas para los ojos (valor alfabético).
ESCENARIO DE ÉXITO: • Se edita el código proporcionado por el departamento de Automática, escribiendo líneas de código que permitan discretizar los datos. • Se definen posiciones absolutas para la cara y ojos • El programa devuelve dichas posiciones

Tabla 66: Caso de uso 001

CU-002: Definir actividades
ACTOR: Investigador
OBJETIVO: Definir un modelo jerárquico de actividades.
PRECONDICIONES: • El investigador debe conocer el dominio de conducción. • Conocer que datos (variables) se pueden extraer del simulador.
POSTCONDICIONES: • Las actividades quedan definidas
ESCENARIO DE ÉXITO: • Se estudian los datos que se pueden extraer del simulador • Se definen los pasos que componen cada actividad. • Se definen las relaciones de inclusión existentes entre maniobras, tareas, acciones y eventos atómicos.

Tabla 67: Caso de uso 002

CU-003: Definir escenarios
ACTOR: Investigador
OBJETIVO: Definir los escenarios ideales para cada actividad.
PRECONDICIONES: • Conocer dominio de conducción. • Estudiar el lenguaje de scripting del simulador. • Estudiar la tipología de los escenarios definidos de serie en el simulador. • Estudiar que datos se quieren extraer del escenario
POSTCONDICIONES: • El escenario queda definido. • El investigador (o usuario) puede realizar una actividad en el escenario.
ESCENARIO DE ÉXITO: • Definir escenario mediante el lenguaje de script.

Tabla 68: Caso de uso 003

CU-004: Realizar actividad
ACTOR: Investigador
OBJETIVO: Realizar las actividades ideales definidas sobre los escenarios definidos.
PRECONDICIONES: • Haber definido las actividades ideales. • Haber definido los escenarios asociados a las actividades.
POSTCONDICIONES: • Se realiza la actividad, siguiendo los pasos definidos, sobre el simulador
ESCENARIO DE ÉXITO: • Encender equipo conectado al simulador. • Iniciar software del simulador. • Cargar escenario definido. • Sentarse en el simulador para que la cámara interior pueda grabar la cara y los ojos. • Iniciar simulación. • Realizar actividad que se desee.

Tabla 69: Caso de uso 004

CU-005: Generar conjunto de datos
ACTOR: Investigador
OBJETIVO: Generar un fichero log (conjunto de datos entrenamiento) al realizar una actividad.
PRECONDICIONES: • Incluir el código necesario en el escenario para poder generar el escenario. • Estudiar qué variables extraer de la simulación.
POSTCONDICIONES: • Se genera un fichero log, denominado conjunto de entrenamiento, compuesto por instancias o filas que contienen los valores de determinadas variables en diferentes instantes secuenciales de tiempo.
ESCENARIO DE ÉXITO: • Encender equipo conectado al simulador. • Iniciar software del simulador. • Cargar escenario definido. • Sentarse en el simulador para que la cámara interior pueda grabar la cara y los ojos. • Iniciar simulación. • Realizar actividad que se desee.

Tabla 70: Caso de uso 005

CU-006: Etiquetar conjunto de entrenamiento
ACTOR: Investigador
OBJETIVO: Etiquetar el conjunto de entrenamiento con acciones atómicas.
PRECONDICIONES: • Fichero log generado. • Acciones atómicas definidas
POSTCONDICIONES: • Se etiqueta un conjunto de entrenamiento. Cada instancia se etiqueta con una o varias clases, ya que se pueden dar acciones concurrentes.
ESCENARIO DE ÉXITO: • Encender equipo conectado al simulador. • Iniciar software del simulador. • Cargar escenario definido. • Ejecutar el programa que etiqueta eventos automáticos

Tabla 71: Caso de uso 006

CU-007: Aplicar trie
ACTOR: Investigador
OBJETIVO: Aplicar un trie sobre el conjunto de entrenamiento etiquetado para generar un primer modelo.
PRECONDICIONES: • Conjunto de entrenamiento generado y etiquetado
POSTCONDICIONES: • Se crea un modelo. Dado un nuevo conjunto de entrenamiento, el sistema es capaz de etiquetarlo correctamente.
ESCENARIO DE ÉXITO: • Aplicar un trie sobre fichero de datos • Obtener secuencias más relevantes. • Modelo generado.

Tabla 72: Caso de uso 008

4.4.2. Descripción gráfica casos de uso

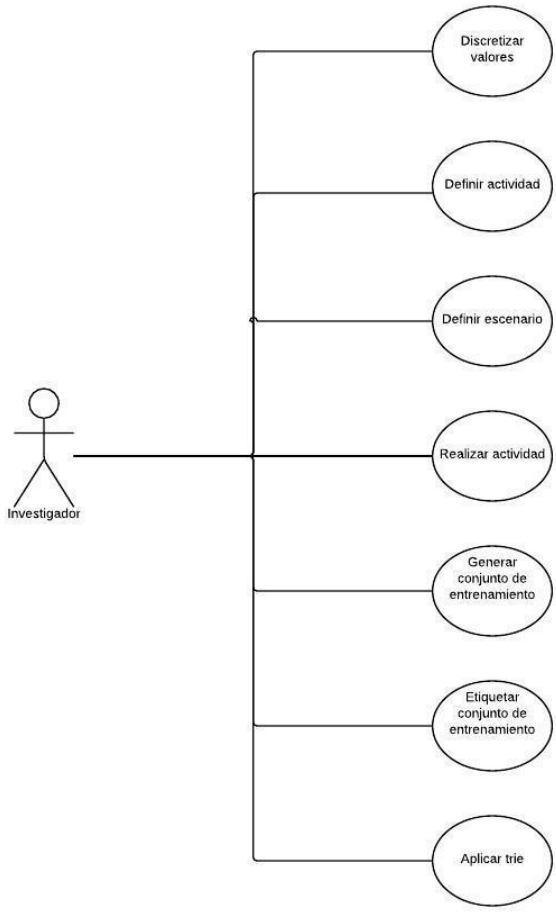

Figura 45: Diagrama casos de uso

4.5. Arquitectura y diseño del sistema

En este apartado se presenta la arquitectura del sistema. El sistema está formado por varios módulos o partes, cada uno de los cuáles desempeña una función dentro del sistema general. Además, cada módulo consta de una serie de componentes que realizan diversas tareas.

Los principales módulos que se distinguen son estos cuatro: módulo simulador, módulo cámara, módulo generación de ficheros y módulo de procesado y etiquetado. En los siguientes subapartados se explica con detalle cada uno de éstos módulos, haciendo hincapié en los componentes que componen cada uno de los módulos. A continuación, se presenta un esquema general de la arquitectura (ver Figura 46) del sistema dividida en módulos, así como la interacción de cada uno de ellos:

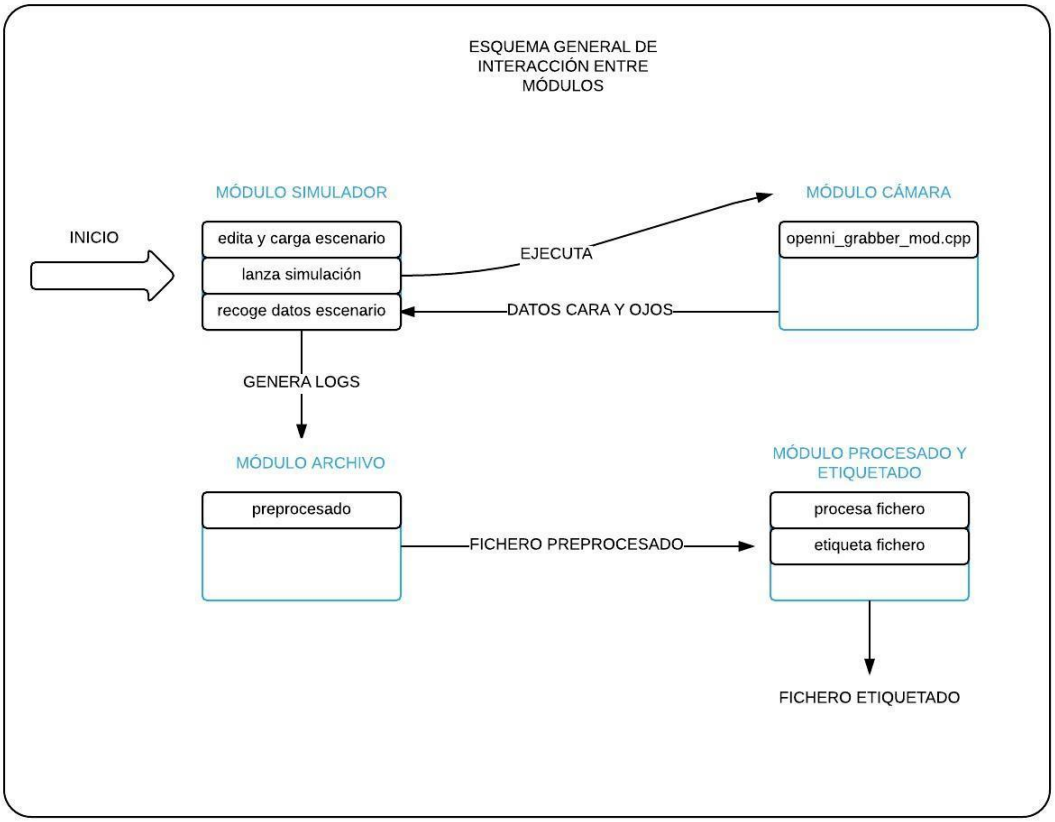

Figura 46: Arquitectura del sistema

4.5.1. Módulo simulador

El primer módulo del sistema (ver Figura 47), y el más importante, es el simulador de coche STISIM. Este simulador consta de los siguientes elementos: tres monitores LCD, un asiento, un volante, unos pedales, un ordenador, un monitor y, lo más importante, un potente software instalado en el ordenador que permite simular un entorno en el cual el usuario pueda conducir. Tanto el volante como los pedales se pueden considerar como sensores del

simulador, ya que se puede registrar en un fichero tanto el ángulo de giro del volante, como el nivel de pisada de alguno de los pedales.

A continuación se muestra el esquema del módulo del simulador:

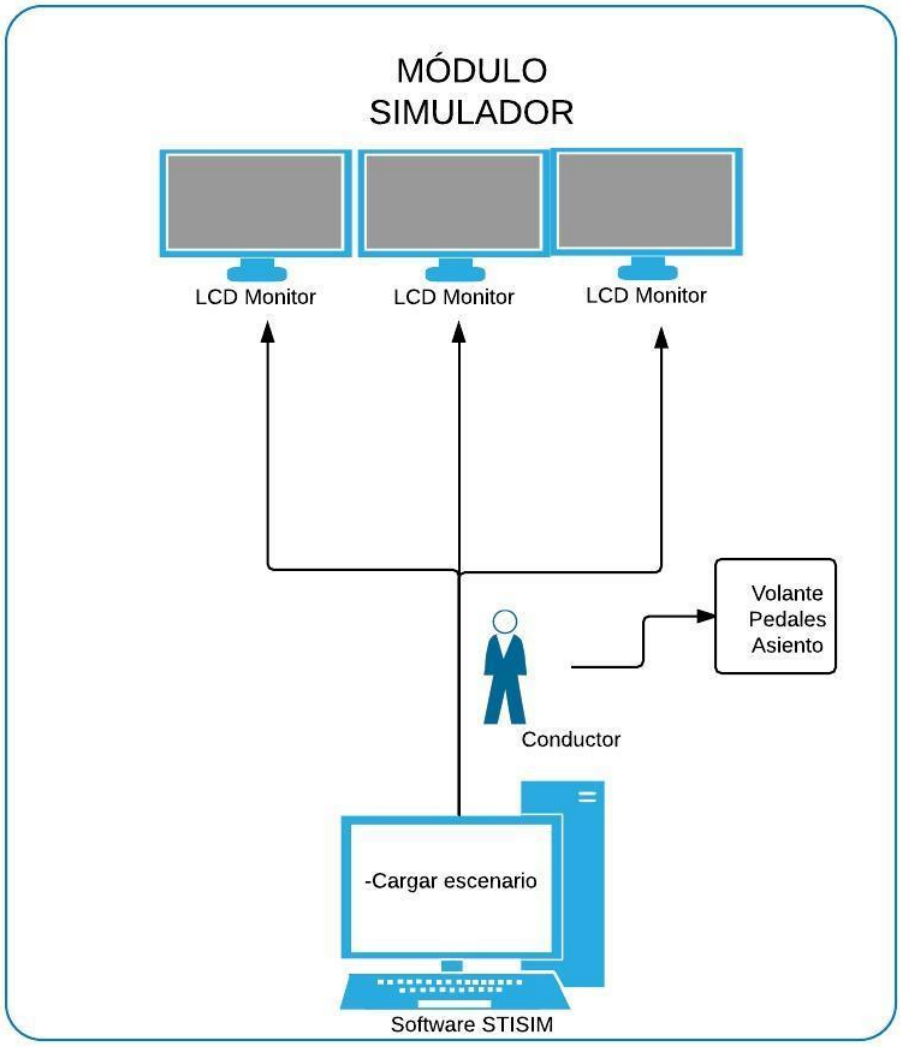

Figura 47: Módulo simulador

A continuación se describen los elementos que componen el módulo simulador, indicando en cada uno de ellos cuál es su funcionalidad dentro del sistema:

- **Monitores LCD:** estos elementos se encargan de simular la visión del conductor, es decir, el usuario observa en ellos todo lo que un conductor ve a través de la luna delantera de su coche. Se encuentran conectados mediante cables HDMI al computador principal donde se instala el software del simulador (ver Figura 48).

Figura 48: LCD

- **Asiento:** este elemento se encarga de simularle al conductor el asiento de un vehículo. No tiene más funcionalidad dentro del sistema.

- **Volante, pedales y palanca de cambios:** estos elementos se describen de manera conjunta ya que se utilizan con el mismo fin: sensores. Se comunican con el computador mediante cables. Además, mediante el software del simulador se utilizan como sensores: cada vez que el conductor mueve el volante o pisa el acelerador (por ejemplo), esta información se registra en el computador. Actúan como sensores fijos (ver *2.2. Reconocimiento basado en sensores)*.

- **Computador principal:** este elemento permite instalar en él el elemento más importante del módulo: el software STISIM. En él, además de instalar el software, se conectan el resto de componentes.

- **Software STISIM:** tal como se ha nombrado antes, es el núcleo de éste módulo y del sistema en general. Mediante el software, se puede editar y cargar un escenario, el cual se proyectará en los monitores. Una vez cargado el escenario, se lanza la aplicación y el usuario (conductor) puede interactuar con el sistema utilizando los pedales, volante o palanca de cambios, pudiendo realizar una simulación de conducción.

4.5.2. Módulo cámara

El segundo módulo del sistema es un computador que tiene instalada la aplicación de detección de cara y ojos del conductor (ver Figura 49). Este módulo se conecta al simulador mediante un socket, utilizando la arquitectura cliente-servidor (este módulo actúa como cliente).

Además de los elementos ya mencionados, se conecta una cámara Kinect al ordenador mediante un cable. Este es un elemento fundamental, ya que permite detectar la cara y los ojos del conductor mediante la ejecución de la aplicación correspondiente. La cámara se debe de situar en una posición que enfoque a la cara del conductor.

Éste módulo es sumamente importante, ya que lanza la aplicación de detección de ojos y cara desarrollada por el departamento de Automática de la UC3M.

A continuación se muestra un esquema general del módulo:

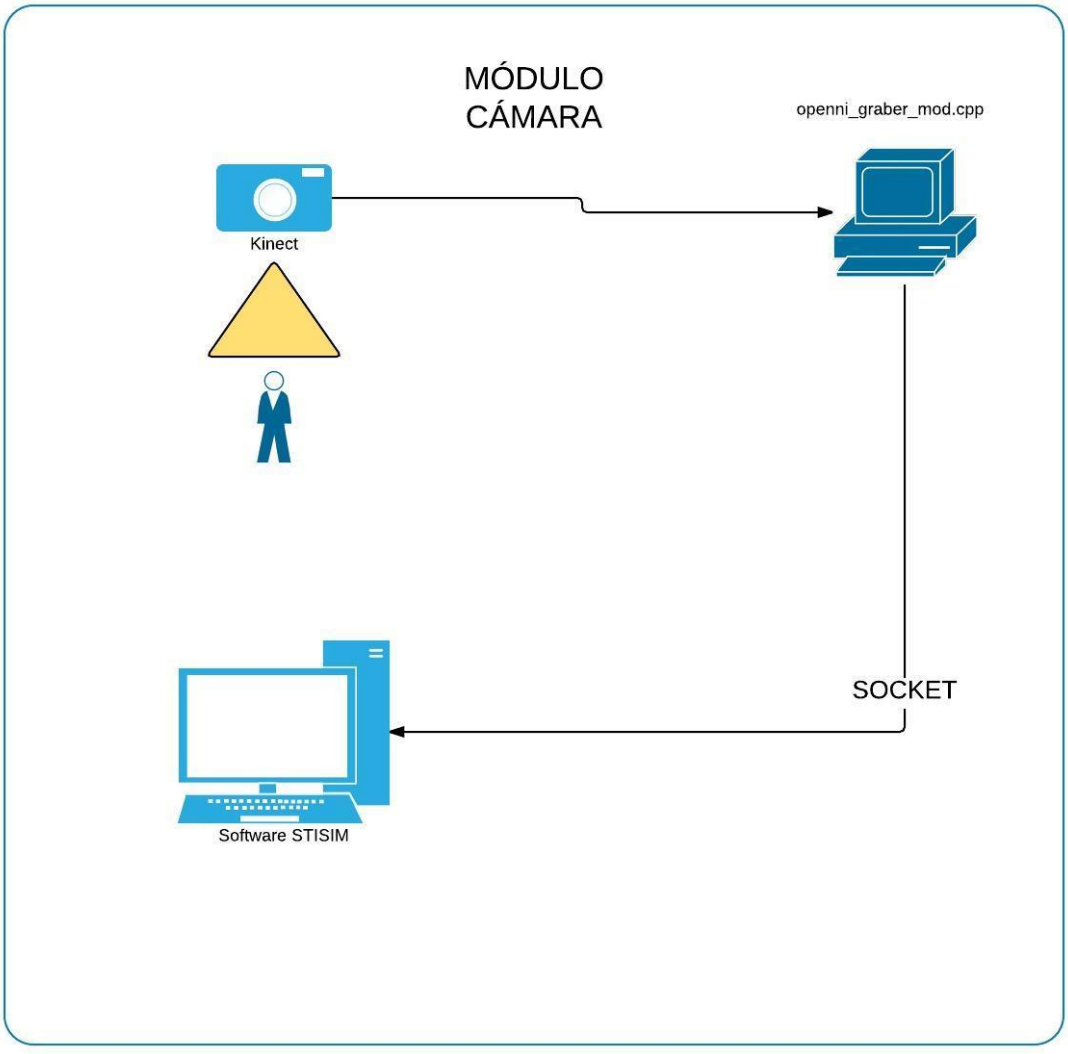

Figura 49: Módulo cámara

A continuación se describen los elementos que componen el módulo cámara, indicando en cada uno de ellos cuál es su funcionalidad dentro del sistema:

- **Kinect:** elemento fundamental dentro de este módulo y dentro del sistema en general. Su funcionalidad reside en grabar la cara y ojos del conductor, a fin de registrar estos datos y poder razonar con ellos posteriormente acerca de a dónde está mirando el conductor. Se encuentra conectada a un ordenador mediante un cable.

- **Aplicación detección de cara y ojos:** se trata de una aplicación instalada en un computador que permite detectar la posición de la cara y ojos del conductor mediante la cámara. Se encuentra conectada al simulador mediante un socket en una arquitectura cliente-servidor, en el que la aplicación se lanza automáticamente cada vez que se inicia una simulación. La aplicación retorna cada instante de tiempo (cada 0.033 segundos) la posición de la cara y ojos del conductor.

- **Software STISIM:** el simulador también forma parte de este módulo, ya que se conecta al ordenador donde se instala la aplicación de detección de ojos mediante un socket.

4.5.3. Módulo generación archivos

El tercer módulo del sistema (ver Figura 51) es el módulo de generación de archivos. Mientras que los otros dos módulos se encargan de generar y cargar los escenarios, y de obtener la posición de la cara y ojos en cada instante de tiempo, este módulo se encarga de generar los ficheros logs.

Para generar los ficheros logs, este módulo toma como entrada la información proporcionada por los sensores del módulo simulador y la información proporcionada por el módulo cámara, generando dos ficheros de log diferentes: uno que contiene la información de los sensores y otro que contiene la información de la cámara. Por último, realiza un pre procesado manual en ambos, eliminando información irrelevante, dejándolos listos para combinarse en una etapa posterior.

Este módulo se lanza de manera automática al iniciar el simulador, siempre que se haya indicado de manera explícita en la definición del escenario.

A continuación se muestra un esquema del presente módulo:

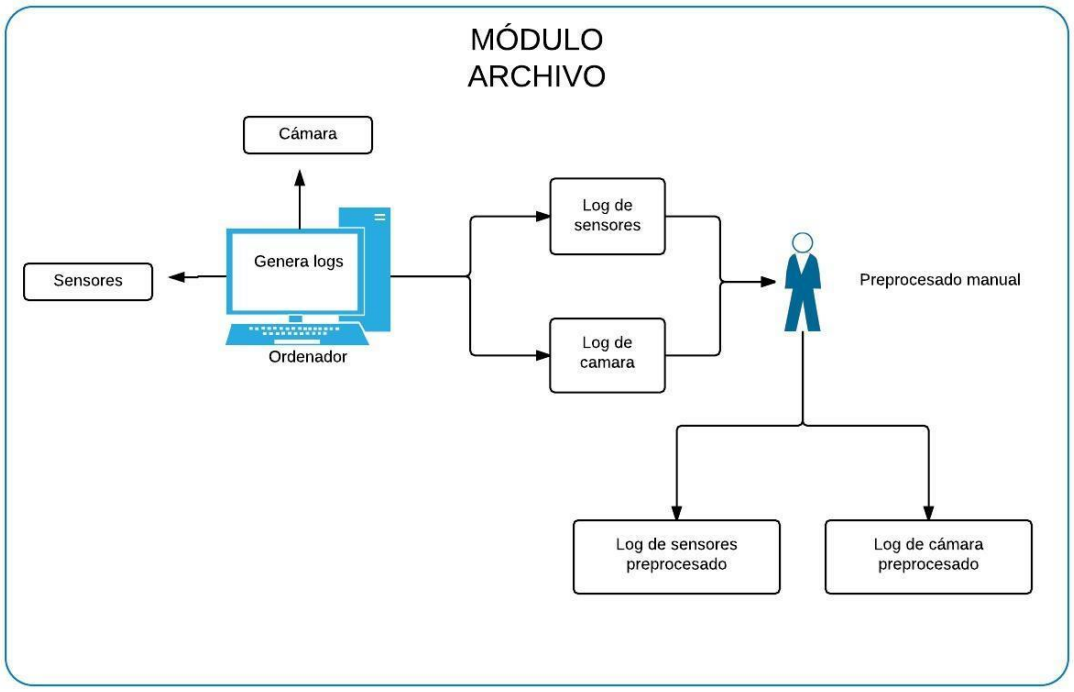

Figura 50: Módulo archivo

A continuación se describen los elementos que componen este módulo:

- **Ordenador:** este es ser el mismo en el que se instala el software del simulador. Genera los logs del simulador (a partir de los sensores y de la información de la simulación mediante el lenguaje de script del STISIM) y de la cámara.

- **Investigador:** el investigador pre procesa los ficheros log de manera manual, eliminando la información irrelevante que se genera en el simulador y generando dos ficheros pre procesados.

4.5.4. Módulo procesado y etiquetado

El cuarto módulo del sistema se encarga de procesar y etiquetar los ficheros (ver Figura 51) resultantes del "módulo archivo", generando un único conjunto de entrenamiento. El procesado consta de varias etapas, explicadas posteriormente. El etiquetado consiste en etiquetar cada fila con uno o varios eventos atómicos (pueden ser concurrentes), y en etiquetar cada fila con una posición que indique hacia donde está mirando el conductor. Por lo tanto, se tienen dos tipos de clase: clase que indica la/s acción/es atómica/s y clase que indica hacia donde mira el conductor.

A continuación se presenta un esquema gráfico del módulo:

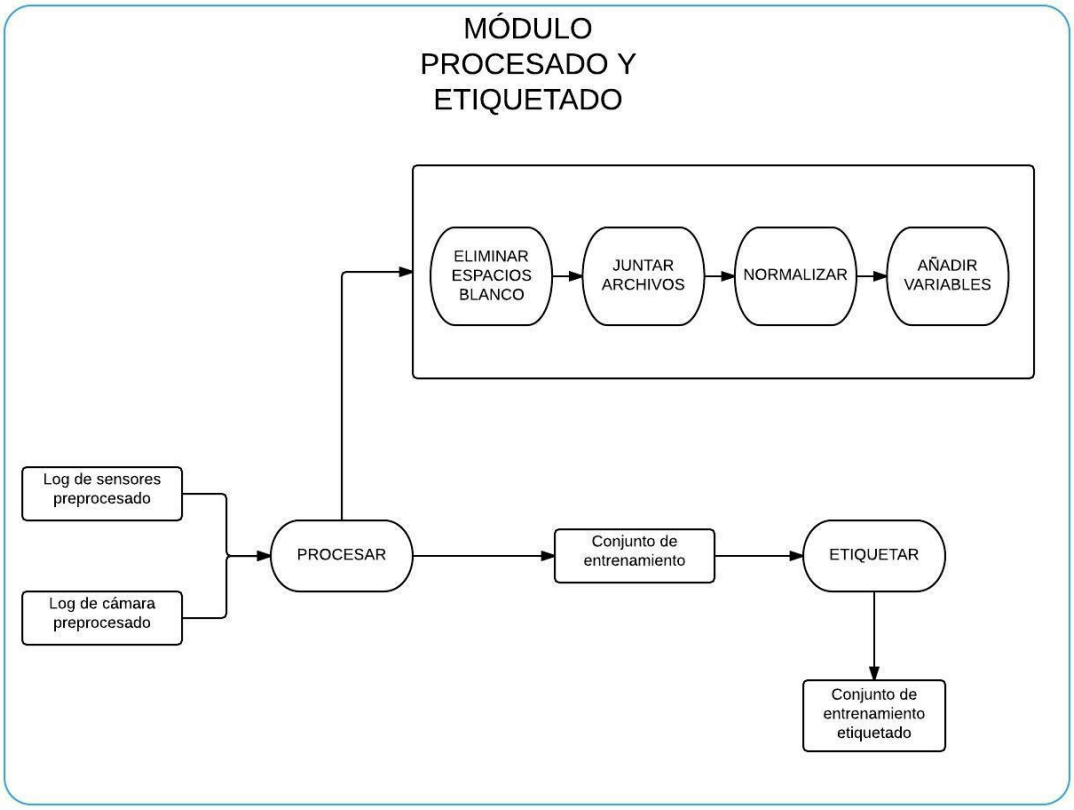

Figura 51: Módulo procesado y etiquetado

El único componente de este módulo es la aplicación Java que se lanza en un ordenador y que realiza todo el proceso mostrado en la ilustración.

Capítulo 4: Implementación del Sistema y Experimentación

5. Implementación del sistema y experimentación

En esta sección se describe con sumo detalle el procesamiento, transformación y razonamiento con los logs generados. Así mismo, se describe con detalle el programa que permite realizar dicho proceso.

En la primera parte del capítulo se describen las consideraciones previas que se han establecido antes de realizar la implementación del sistema.

En la segunda parte del capítulo se describe el proceso llevado a cabo para transformar los ficheros logs generados al realizar las actividades en conjuntos de entrenamiento etiquetados. Para ello, se incluye un diagrama de flujo de programa.

En la tercera parte del capítulo, se detalla la implementación del código, realizando un diagrama de clases.

En la cuarta y última parte del capítulo, se aplica un trie descriptivo sobre los diferentes ficheros de datos que representan algunas tareas, a fin de realizar una primera aproximación del modelo práctico.

5.1. Consideraciones previas

Antes de comenzar la implementación del sistema, es necesario establecer una serie de consideraciones previas. El modelo de actividades propuesto en el capítulo 3.2. Modelo jerárquico de actividades, es un modelo teórico e ideal, por lo que su implementación total es una tarea que se antoja difícil. En concreto, se determinan las siguientes consideraciones previas:

- La implementación del sistema constituye una primera aproximación para la detección de patrones de comportamiento del conductor. Este proyecto basa su “núcleo” en la definición teórica del modelo de actividades, con el fin de proporcionar una base teórica a la implementación práctica. Por ello, esta implementación práctica es susceptible de numerosas mejoras o ampliaciones.

- Para cada una de las actividades realizadas en el simulador, se ha estudiado qué variables del simulador son las necesarias. En caso de implementar este sistema en tiempo real, el simulador debería extraer todas las variables posibles del simulador y realizar un filtrado automático (líneas futuras del proyecto).

- Los modelos ideales y teóricos de actividades definidos en el capítulo 3.2. Modelo jerárquico de actividades son bastante complejos en el ámbito práctico. El principal problema de estos modelos teóricos ideales es la existencia de eventos atómicos concurrentes, es decir, un conductor puede estar realizando dos eventos atómicos de manera simultánea, como por ejemplo, pisando el acelerador y moviendo el volante a la derecha. En el modelo teórico, en cambio, se asume que estos eventos ocurren de

manera secuencial y que el vehículo va transitando entre un conjunto de estados a medida que van ocurriendo los eventos concurrentes. Eventos concurrentes implican estados concurrentes, es decir, un vehículo puede estar en varios estados de manera simultáneamente, lo que complica aún más el proceso de detección de patrones.

- Los ficheros log de cada una de las actividades incluyen estados del vehículo ("*gear*" o "*speed*") de manera explícita, por lo tanto, estos estados son directamente observables del simulador y existe una columna en cada valor que representa cada uno de estos estados. Sin embargo, existen determinados estados, como "*huecoAparcar = TRUE"* que no son directamente observables, si no que incluyen una serie de cálculos para conocer si se cumple o no la condición. Dada la complejidad de esto, se ha decidido no incluir una columna en los ficheros de *log* que incluya si se da este estado o no.

- Dentro de un escenario, sólo se realiza una única vez cada una de las tareas/maniobras realizadas. Por ejemplo, para la maniobra de "Adelantar", únicamente se repite una vez la secuencia de eventos/acciones/tareas que componen dicha maniobra. Esto es así porque se pretende que el log generado en dicha maniobra sea "ideal" y "único", es decir, que represente lo más fielmente posible a la secuencia ideal de acciones que realiza un conductor cuando adelanta a un único vehículo.

- Cada escenario se ha modificado acorde a las necesidades de las actividades. Por ejemplo, para la maniobra de "Ceder paso de peatones", se ha introducido un paso de peatones en un escenario, con un peatón cruzando en el momento en el que el vehículo llega al paso de peatones.

5.2. Procesado y etiquetado de datos

En este apartado se describe el proceso llevado a cabo para transformar los ficheros logs generados al realizas las actividades en conjuntos de entrenamiento etiquetados (ver Figura 58). Para cada maniobra y tarea definida, se generan dos ficheros de logs: un fichero de log con las variables definidas en los apartados 3.2.3. Tareas y 3.2.4. Maniobras, y un fichero log con la posición de cara y ojos del conductor. Ambos ficheros se han generado de manera sincronizada, es decir, cada instancia de un fichero corresponde en el tiempo a una instancia del otro fichero.

El objetivo es, para cada tarea, generar un conjunto de entrenamiento etiquetado con los eventos atómicos que se realizan en cada instante de tiempo. El conjunto de esos eventos atómicos definen una tarea, por lo que se genera otro fichero que contenga esos eventos atómicos: ese fichero representará una tarea. Posteriormente, mediante la aplicación de un trie, se genera un modelo de tareas que permite identificar tareas dados unos eventos atómicos.

El proceso es el siguiente:

1. **Pre procesado manual:** los ficheros logs del simulador se generan con información innecesaria, por lo que es necesario eliminar manualmente esta información (ver Figura 52: Pre procesado manual). Esta información es la definición del escenario, metadatos como la fecha de creación, número de licencia, configuración, etc.

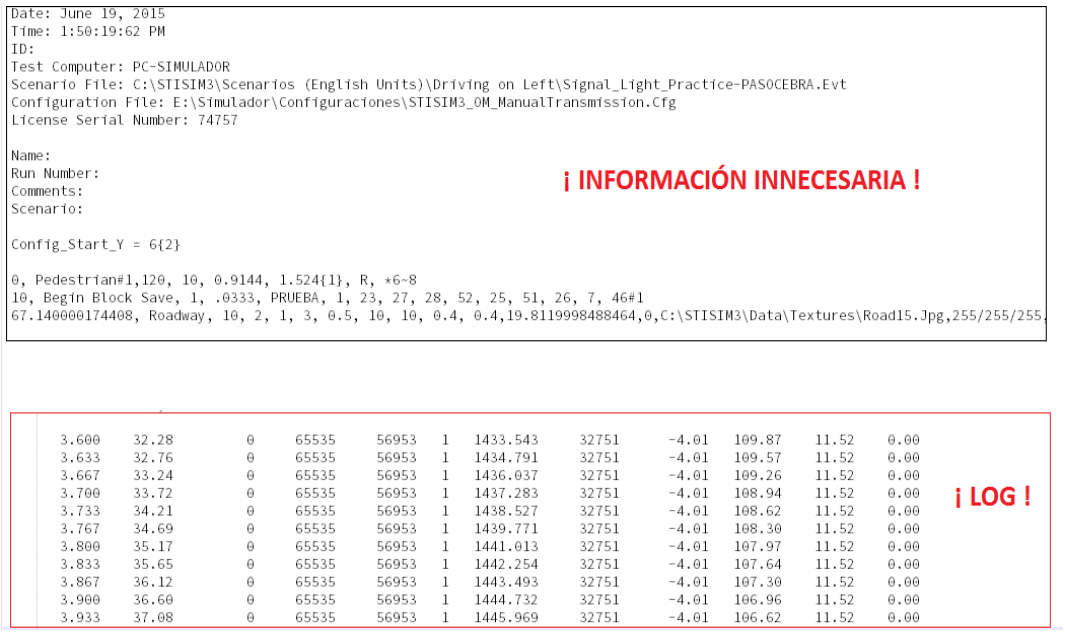

Figura 52: Pre procesado manual

2. **Eliminación de espacios:** se eliminan los espacios al principio y al final del fichero (ver Figura 53: Eliminación de espacios . Además, con el objetivo de crear todos los ficheros con un formato común, se sustituyen los espacios entre variables por ";" para generar un fichero separados por ";" (denominado *fichero CSV*).

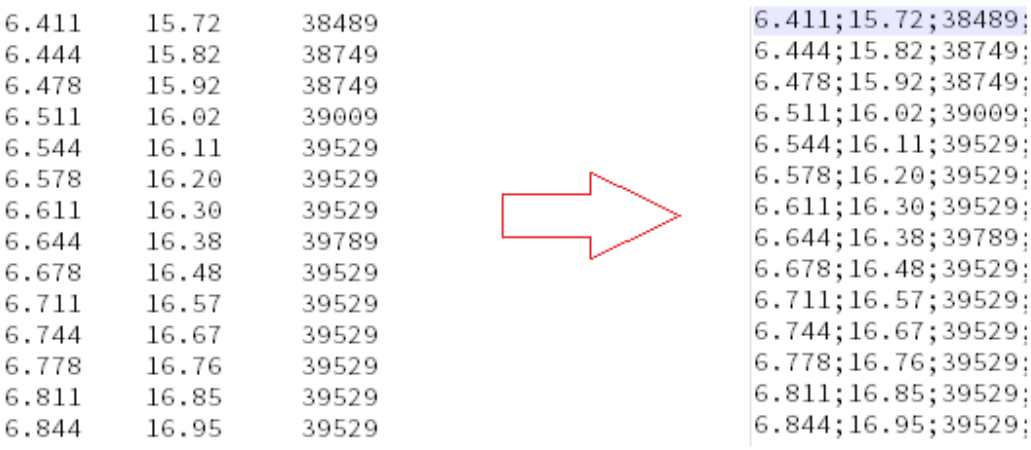

Figura 53: Eliminación de espacios

3. **Combinación de ficheros:** se combinan cada par de ficheros de simulador y ficheros de cámara en un único fichero, añadiendo la posición de la cara y de los ojos de cada instante al final de cada instancia del fichero del simulador (ver Figura 54).

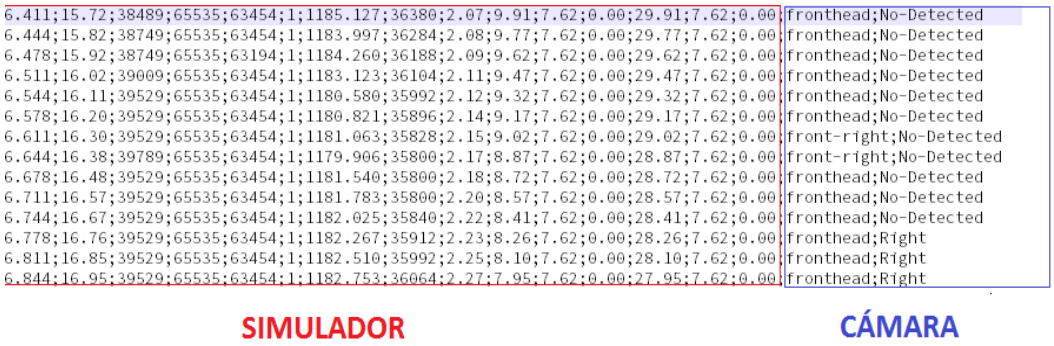

Figura 54: Combinación ficheros

4. **Normalizar datos:** se normalizan entre 0 y 1 las variables que indican el nivel de pisada de aceleración, freno y embrague, así como la dirección del volante, siendo "0" no pisado y "1" pisado al máximo, y "0" el volante totalmente a la derecha y "1" totalmente a la izquierda (ver Figura 55).

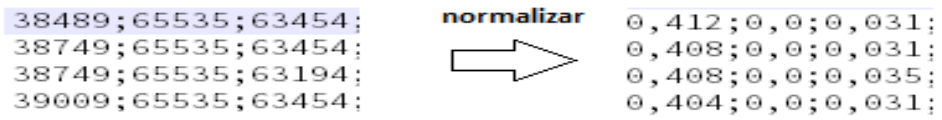

Figura 55: Normalizar

5. **Almacenar en memoria el fichero:** para poder razonar y operar con los datos, es necesario guardarlos en memoria en alguna estructura de datos, como puede ser un array.

6. **Añadir variables:** en esta fase se añaden variables que se consideran importantes y no se podían extraer directamente del simulador. Una de estas variables, por ejemplo, es la diferencia de velocidad entre un instante de tiempo y otro.

7. **Etiquetado de tareas:** para cada fichero de cada tarea, se razona con los datos guardados en memoria y se etiquetan con los eventos y estados atómicos reconocidos, los cuales son definidos en el apartado 3.2.1. Eventos atómicos., generando ficheros etiquetados (ver Figura 56). Para ello, se compara cada instante con el siguiente. Por ejemplo, si la marcha en el instante t=5 es "2" y en el instante t=6 es "3", entonces en el instante t=5 se ha producido el evento atómico "changingGearUp".

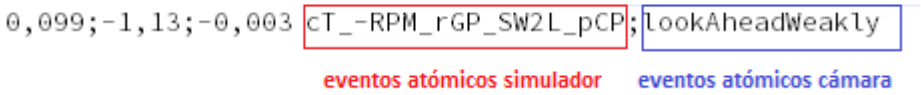

Figura 56: Etiquetado eventos atómicos

8. **Generación modelo de tareas:** dado un fichero de tarea etiquetado, se genera un fichero de tarea que contenga todos los eventos atómicos separados por comas(ver Figura 57) y, por último, la clase de ese fichero (el nombre de la tarea). Este fichero se utiliza para aplicar un *trie* sobre él que permita describir a la tarea en sí, obteniéndose los eventos atómicos más relevantes de la tarea, es decir, aquellos que mejor representan a la tarea.

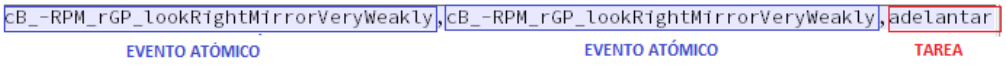

Figura 57: Generación tareas

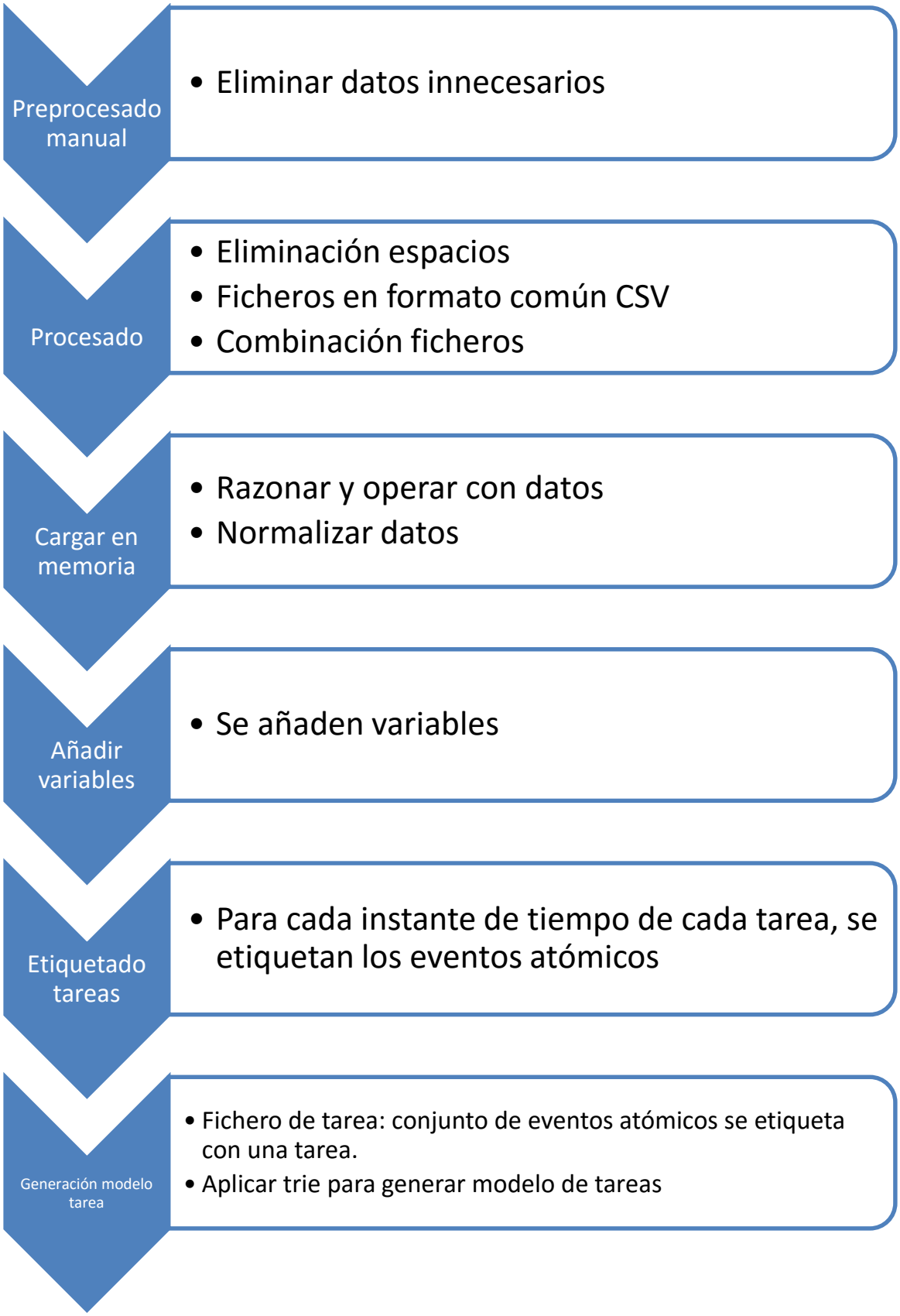

Figura 58: Diagrama de flujo

5.3. Codificación

Para realizar toda la funcionalidad descrita en el apartado anterior, se implementa un programa en el lenguaje Java. Se ha decidido utilizar este lenguaje por su facilidad para leer y escribir ficheros mediante las clases *File, FileReader, FileWriter, PrintWriter y BufferedReader*. Además, es un lenguaje donde el investigador se siente muy cómodo, dado su avanzado nivel de programación en dicho lenguaje.

A continuación, se muestra un diagrama de clases del programa (ver Figura 59) :

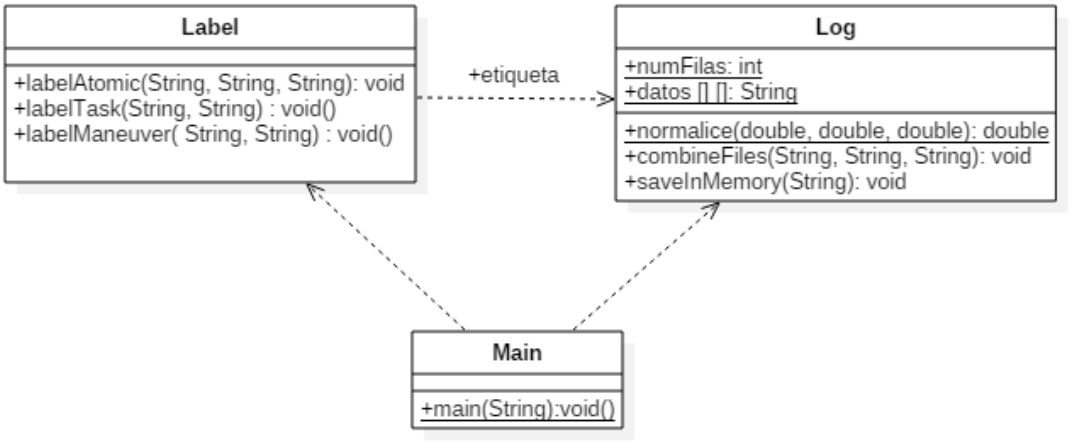

Figura 59: Diagrama de clases

Como se observa en la figura anterior, la clase *Label* depende de la clase *Log*, ya que los métodos de *Label* necesitan utilizar los datos del fichero que se genera en el método *combineFiles* de la clase *Log*. Además, la clase *Main* depende de *Label y Log*, ya que en ella se ejecuta el método *main* del programa.

A continuación se describen todas las clases, métodos y atributos del presente proyecto.

5.3.1. Clase Log

ATRIBUTOS

Los atributos de la clase son:

ATRIBUTOS
• ***static int numFilas:*** almacena el número de filas del fichero a generar. Es estático porque se le da valor al leer el fichero de log del simulador, ya que el número de filas del fichero generado coincidirá con el número de filas del log generado en el simulador.
• ***static String datos[][]****:* matriz bidimensional que almacena el fichero generado. Variable estática (global) porque se pasa como parámetro a la clase *Label*.

Tabla 73: Atributos clase LOG

MÉTODOS

Los métodos de la clase son:

public statitic double normalice(double max, double min, double valor)	
Entrada: *-max:* valor máximo (0 en este caso) *-min:* valor mínimo (65535 en este caso*)* *-valor:* valor a normalizar	***Salida****:* -valor normalizado entre 0 y 1
Algoritmo: El método normaliza el valor deseado entre 0 y 1. Para ello, divide la resta de *valor-min* entre *max-min* y, al resultado, le aplica valor absoluto mediante la función *abs* de la clase *Math.*	

Tabla 74: Método normalice

public void combineFiles(String rutaDat, String rutaCSV, String ficheroProcesado)	
Descripción: Este método combina los ficheros de log del simulador y cámara en uno sólo. Además, procesa dicho fichero, eliminando espacios en blanco y convirtiéndolo en formato CSV (archivo separado por comas).	
Entrada: *-rutaDat:* ruta del fichero de log del simulador *-rutaCSV:* ruta del fichero de log de la cámara *-ficheroProcesado:* ruta del fichero a generar	***Salida****:*
Algoritmo: 1. Apertura ficheros DAT y CSV 2. Lectura ficheros DAT y CSV 3. Para cada línea del fichero DAT: 3.1. Se concatena cada línea del fichero DAT con cada línea del fichero CSV 3.2.Procesado de línea: -Eliminación espacios al principio y al final mediante *trim()* -Sustitución de espacios intermedios por ";" (formato CSV) 3.3. Escritura de línea en *ficheroProcesado* mediante *println()* 3.4. Volcar *buffer* mediante *flush()* para asegurarnos la escritura total 3.5. numFilas= numFilas+1 4. Cierre de ficheros para evitar errores	

Tabla 75: Método combineFiles

public void saveInMemory(String ruta)	
Descripción: Este método lee el fichero pasado por parámetro y lo almacena en la matriz bidimensional *datos*. Para poder razonar y operar con los datos de un fichero es necesario cargarlo en memoria, por lo que este método es fundamental.	
Entrada: *-ruta:* ruta del fichero de log combinado anteriormente.	***Salida****: El fichero se carga en memoria.*
Algoritmo: 1. Apertura fichero a guardar en memoria. 2. Lectura fichero mediante *readLine() (método de la clase BufferedReader).* 3. Reservar memoria para *datos*, con tantas filas como *numFilas* 3. Cada línea del fichero se almacena en cada fila de la matriz, ocupando tantas columnas como variables tenga el fichero. 4. Para las variables *GasPedal, BrakePedal, ClutchPedal y SteeringWheelAngle:* -Normalizar datos entre 0 y 1, siendo 0 mínimo y 1 máximo. 5. Reiniciar *numFilas* a 0.	

Tabla 76: Método saveInMemor

5.3.2. Clase Label

ATRIBUTOS

Esta clase no tiene atributos.

MÉTODOS

public void labelAtomic(String datos[][], String ruta, String cabecera)	
Descripción: Este método lee los datos almacenados en la matriz *datos*, razona con ellos y etiqueta cada una de las instancias del fichero *ruta* con eventos atómicos. Para ello, compara cada instancia con la siguiente, es decir, cada instante de tiempo con el inmediatamente posterior.	
Entrada: *-datos:* estructura de memoria (matriz bidimensional) donde está almacenado el fichero generado en la clase *Log.* *-ruta:* ruta del fichero etiquetado a generar *-cabecera:* cabecera de cada fichero a generar, ya que cada actividad realizada tiene unas variables diferentes.	***Salida:*** **-**Se genera un fichero etiquetado con eventos atómicos.
Algoritmo: 1. Apertura y lectura fichero de cabecera (*cabecera)*. En él se encuentran los nombres de las variables contenidas en los datos. Se almacena en estructura de datos auxiliar. 2. Obtención posiciones de las variables dentro del array *datos*. Para poder operar y razonar con las variables, es necesario conocer en qué columna se encuentra cada variable: 2.1. Se recorre array de cabecera. -Si arrayCabecera[i] == Variable ENTONCES posicionVariable=i 3. Se recorre matriz de datos. Para cada fila: 3.1. Se escribe la fila almacenada (se escriben todas las variables) en *ruta* 3.2. Se añaden a la fila las variables *diferencia_velocidad, diferencia_rpm, diferencia_sw* 3.3. Se etiqueta cada fila con eventos atómicos del simulador -Se compara cada fila con la siguiente. Ejemplo: SI velocidad en t=0 < velocidad en t=1 ENTONCES eventoAtomico= carThrootle. 3.4. Se etiqueta cada fila con eventos atómicos de la cámara -Se aplica el razonamiento descrito en *4.2.1. EVENTOS ATÓMICOS* 4. Se cierra el fichero *ruta*	

Tabla 77: LabelAtomic

public void labelTask(String fileIn, String fileOut)	
Descripción: Este método lee un fichero etiquetado con eventos atómicos y genera un fichero de una sola línea que contiene todos los eventos atómicos del fichero de entrada separados por comas. Finalmente, etiqueta dicha línea con una clase, en este caso, con la tarea realizada.	
Entrada: *-fileIn:* Fichero de log etiquetado con eventos atómicos *-fileOut:* Fichero a generar	***Salida****:* **-**Se genera un fichero que contiene una única línea con todos los eventos atómicos del fichero de entrada separados por comas.
Algoritmo: 1. Abrir fichero *fileIn* 2. Lectura de cabecera (primera línea) para obtener posiciones de evento atómico de simulador (*AtomicAction)* y de evento atómico de cámara *(look).* 3. Para cada línea: -Almacenar linea en array auxiliar (cada variable en una posición) -Concatenar en una cadena los valores de *AtomicAction y look*. -Si cadena EMPIEZA POR "_" ENTONCES eliminar "_" -Escribir línea -Etiquetar línea con nombre del fichero *fileIn* (coincide con el nombre de la tarea)	

Tabla 78: LabelTask

public void labelManeuveur(String fileIn, String fileOut)	
Descripción: Este método lee un fichero etiquetado con eventos atómicos y genera un fichero de una sola línea que contiene todos los eventos atómicos del fichero de entrada separados por comas..	
Entrada: *-fileIn:* Fichero de log etiquetado con eventos atómicos *-fileOut:* Fichero a generar	***Salida****:* **-**Se genera un fichero que contiene una única línea con todos los eventos atómicos del fichero de entrada separados por comas.
Algoritmo: Mismo funcionamiento que *labelTask*, excepto que en este método no se etiqueta la maniobra.	

Tabla 79: LabelManeuver

5.3.3. Clase Main

ATRIBUTOS

Esta clase no tiene atributos

MÉTODOS

public void static void main(String [] args)	
Descripción: Método main del proyecto, es decir, método que se encarga de invocar al resto de métodos y ejecutar todo el código contenido en ellos.	
Entrada:	***Salida:***
Algoritmo: 1. Para cada par de ficheros de log (simulador y cámara) preprocesados: -Invocar *combineFiles* para juntarlos en un solo fichero de log procesado -Invocar *saveInMemory* para cargar en memoria el fichero de log -Invocar *labelAtomic* para etiquetar el fichero con eventos atómicos - Si Actividad == Tarea ENTONCES invocar *labelTask* EN OTRO CASO invocar *labelManeuver* *NOTA: se debe de seguir el orden de invocaciones establecido, ya que la salida de cada método es la entrada del siguiente.*	

Tabla 80: Main

5.4. Resultados obtenidos

En este apartado se detalla la experimentación llevada a cabo en el presente Trabajo Final de Grado, presentando los resultados obtenidos. Como se detalla en apartados anteriores, una vez generado y procesado un fichero de log de una actividad, este fichero es etiquetado, instancia por instancia, con los eventos atómicos detectados en éste. De esta forma, se tiene un conjunto de datos compuesto por los eventos atómicos que representan una tarea. En el siguiente ejemplo (ver Figura 60), se muestra la experimentación realizada al etiquetar eventos atómicos (columnas "*AtomicAction*" y "*look*") para la tarea "Detenerse".

timestamp(s)	speed(km/h)	gaspedal	brakepedal	clutchpedal	gear	rpm	SteeringWheel	head	eyes	SpeedDifference	RPMDifference	swDifference	AtomicAction	look
21,767	122,83	1	0	0	4	3692,015	0,501	up-left	No-Detected	0	0	0	cT_+RPM_pGP	lookAheadWeakly
21,8	122,91	1	0	0	4	3693,378	0,501	up-left	No-Detected	0,079	1,363	0	cT_:RPM_rGP	lookAheadWeakly
21,833	122,97	0,968	0	0	4	3675,238	0,501	up-left	No-Detected	0,06	-18,141	0	cT_:RPM_rGP	lookAheadWeakly
21,867	123,01	0,937	0	0	4	3656,838	0,501	up-left	No-Detected	0,04	-18,4	0	cT_:RPM_rGP	lookAheadWeakly
21,9	123,05	0,914	0	0	4	3643,065	0,501	front-left	No-Detected	0,039	-13,774	0	cT_:RPM_rGP	lookLeftMirrorVeryWeakly
21,933	123,06	0,882	0	0	4	3624,225	0,501	front-left	No-Detected	0,01	-18,841	0	cB_:RPM_rGP	lookLeftMirrorVeryWeakly
21,967	123,05	0,828	0	0	4	3590,504	0,501	up-left	No-Detected	-0,011	-33,721	0	cB_:RPM_rGP	lookAheadWeakly
22	123,01	0,777	0	0	4	3558,792	0,501	up-left	No-Detected	-0,04	-31,712	0	cB_:RPM_rGP	lookAheadWeakly
22,033	122,96	0,726	0	0	4	3526,687	0,501	up-left	No-Detected	-0,051	-32,106	0	cB_:RPM_rGP	lookAheadWeakly
22,067	122,85	0,648	0	0	4	3477,133	0,501	up-left	No-Detected	-0,11	-49,555	0	cB_:RPM_rGP	lookAheadWeakly
22,1	122,7	0,535	0	0	4	3405,071	0,501	up-left	No-Detected	-0,15	-72,062	0	cB_:RPM_rGP	lookAheadWeakly
22,133	122,5	0,432	0	0	4	3338,852	0,501	up-left	No-Detected	-0,201	-66,22	0	cB_:RPM_rGP	lookAheadWeakly
22,167	122,26	0,325	0	0	4	3269,111	0,501	up-left	No-Detected	-0,24	-69,741	0	cB_:RPM_rGP	lookAheadWeakly
22,2	121,95	0,182	0	0	4	3176,437	0,501	up-left	No-Detected	-0,311	-92,674	0	cB_:RPM_rGP	lookAheadWeakly
22,233	121,62	0,083	0	0	4	3110,087	0,501	up-left	No-Detected	-0,33	-66,35	0	cB_:RPM_rGP	lookAheadWeakly
22,267	121,28	0	0	0	4	3053,242	0,501	up-left	No-Detected	-0,341	-56,845	0	cB_:RPM	lookAheadWeakly
22,3	120,95	0	0	0	4	3047,639	0,501	up-left	No-Detected	-0,33	-5,604	0	cB_:RPM	lookAheadWeakly
22,333	120,62	0	0	0	4	3042,207	0,501	up-left	No-Detected	-0,33	-5,433	0	cB_:RPM	lookAheadWeakly
22,367	120,31	0	0	0	4	3036,883	0,501	up-left	No-Detected	-0,311	-5,325	0	cB_:RPM	lookAheadWeakly
22,4	120	0	0	0	4	3031,662	0,501	front-left	No-Detected	-0,311	-5,222	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,433	119,7	0	0	0	4	3026,539	0,501	front-left	No-Detected	-0,3	-5,123	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,467	119,4	0	0	0	4	3021,508	0,501	front-left	No-Detected	-0,3	-5,032	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,5	119,11	0	0	0	4	3016,564	0,501	front-left	No-Detected	-0,291	-4,944	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,533	118,82	0	0	0	4	3011,703	0,501	front-left	No-Detected	-0,291	-4,861	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,567	118,54	0	0	0	4	3006,919	0,501	front-left	No-Detected	-0,28	-4,785	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,6	118,26	0	0	0	4	3002,21	0,501	front-left	No-Detected	-0,281	-4,709	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,633	117,99	0	0	0	4	2997,57	0,501	front-left	No-Detected	-0,271	-4,64	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,667	117,72	0	0	0	4	2992,996	0,501	front-left	No-Detected	-0,27	-4,575	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,7	117,45	0	0	0	4	2988,485	0,501	front-left	No-Detected	-0,27	-4,511	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,733	117,19	0	0	0	4	2984,033	0,501	front-left	No-Detected	-0,261	-4,453	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,767	116,93	0	0	0	4	2979,638	0,501	front-left	No-Detected	-0,26	-4,395	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,8	116,67	0	0	0	4	2975,296	0,501	front-left	No-Detected	-0,261	-4,343	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,833	116,42	0	0	0	4	2971,004	0,501	front-left	No-Detected	-0,25	-4,292	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,867	116,17	0	0	0	4	2966,761	0,501	front-left	No-Detected	-0,25	-4,243	0	cB_:RPM	lookLeftMirrorVeryWeakly
22,9	115,92	0	0	0	4	2962,563	0,501	up-left	No-Detected	-0,25	-4,198	0	cB_:RPM	lookAheadWeakly
22,933	115,68	0	0	0	4	2958,408	0,501	up-left	No-Detected	-0,24	-4,156	0	cB_:RPM_SW2L	lookAheadWeakly
22,967	115,43	0	0	0	4	2954,294	0,5	up-left	No-Detected	-0,25	-4,115	-0,002	cB_:RPM	lookAheadWeakly
23	115,19	0	0	0	4	2950,219	0,5	fronthead-up	No-Detected	-0,241	-4,075	0	cB_:RPM	lookRearMirrorWeakly

Figura 60: Etiquetado ficheros

En la Figura 60, se observa el etiquetado de eventos atómicos en un conjunto de datos. Por ejemplo, para la instancia de *timestamp=22,* se compara esa instancia con la siguiente y, en base a los datos de las siguientes columnas, se realiza el etiquetado automático de los datos. En este ejemplo, se detectan los siguientes eventos atómicos:

- :RPM: reducción de revoluciones. Las revoluciones pasan de ser 3590,5 a ser 3558,7.
- rGP: dejar de pisar acelerador. El nivel de aceleración pasa de 0,82 a 0,77
- cB: el coche frena, es decir, la velocidad es inferior al instante anterior (pasa de 123,01 a 122,96)

Para el etiquetado de eventos atómicos inferidos de las posiciones de la cara y ojos, se razona con los datos de la propia instancia. En este caso, la instancia se etiqueta con el evento "*lookAheadWeakly*", inferido de las columnas "*head*" y "*eyes*".

5.4.1. Aproximación práctica propuesta

El enfoque propuesto para realizar una primera aproximación de un modelo práctico se basa en el descrito en [21].

Partiendo de un fichero de datos asociado a una tarea, se deduce que la secuencia de eventos atómicos reconocidos en dicho fichero representa a la tarea en sí. Una secuencia de eventos atómicos {A1,A2,A3..,AN} es un conjunto de eventos atómicos ordenados en el tiempo.

El objetivo es obtener aquellas secuencias de eventos más relevantes, ya que éstas serán las que describan más fielmente a la tarea en sí. De este modo, en un futuro se podría reconocer una tarea en base a los eventos más relevantes que la componen.

El enfoque propuesto se basa en asumir que las secuencias de eventos más relevantes son aquellas que más se repiten. Esto último ya se realiza en áreas como minería de textos (*text mining)*, donde se detectan las palabras más relevantes de un texto en base a su frecuencia dentro del mismo.

Para obtener la secuencia de eventos atómicos más relevantes de una tarea se utiliza una estructura en forma de árbol denominada *trie,* la cual permite identificar los eventos atómicos más repetidos, así como su dependencia con eventos anteriores y posteriores en el tiempo. Un *trie* es una estructura en forma de árbol en el que los datos se insertan en forma ordenada en nodos, siendo cada nodo una subsecuencia, teniendo todos los nodos un nodo padre denominado *root (ver* Figura 61*)*. Esta estructura se organiza por niveles (profundidad), lo que significa que para llegar a un nodo de nivel 2 es necesario pasar por un nodo de nivel 1.

Dada la secuencia de {ls,date,ls,date,cat}, por ejemplo, se elige una profundidad (por ejemplo, 3) y se generan subsecuencias de ese tamaño: {ls,date,ls}, {date,ls,date}, {ls,date,cat}.

Seguidamente se insertan en el trie las subsecuencias de cada subsecuencia. Por ejemplo, para la primera subsecuencias, se insertan en el trie las siguientes subsecuencias: {ls,date,ls}, {date,ls} y {ls}.

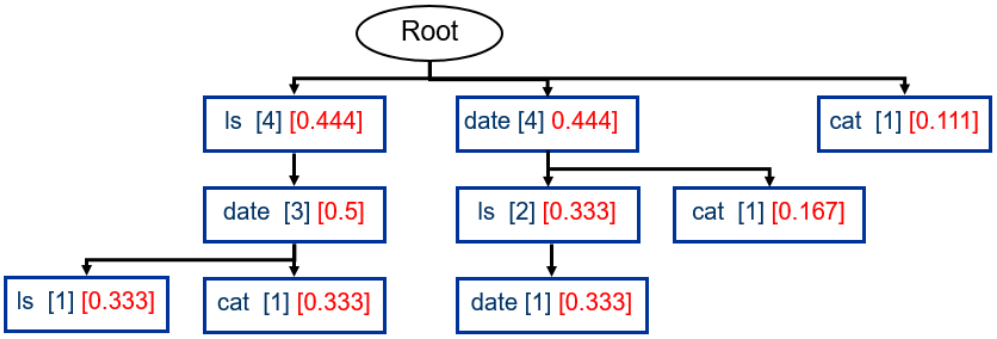

Figura 61: Trie [21]

Cada vez que se inserta una subsecuencia, se aumenta la frecuencia de inserción (*corchete azul).* Por otro lado, el *corchete rojo* indica la relevancia de esa subsecuencia en el modelo final. En el ejemplo, la subsecuencia más relevante es {ls, date}, con un valor de relevancia (*support)* de 0.5.

Con las secuencias de eventos atómicas obtenidas, el objetivo es generar un *trie* descriptivo, es decir, un trie que identifique aquellas secuencias de eventos atómicos que mejor describan a una tarea.

A continuación, se describe el proceso realizado para generar una primera aproximación del modelo de tareas de "Detenerse" y "Arrancar y salir".

5.4.2. Modelo de tareas: detenerse

Para esta primera aproximación se decide estudiar la tarea de "Detenerse". Se recuerda que dicha tarea se basa en los siguientes eventos atómicos (ver Tabla 10):

- *releaseGasPedal (rGP)*
- *pushBreakPedal (pBP)*
- *clutched (pCP)*
- *changingGearDown (G-)*

Además, tal y como se describe en el modelo teórico ideal de actividades, el vehículo debe de pasar por los estados de "*decreasingRPM*" y "*carBraking*".

Es importante observar que en esta tarea no están implicados los eventos atómicos relacionados con la posición de la cara y ojos del conductor, por lo que se decide no incluir estos eventos en el modelo, ya que se incrementaría la complejidad del experimento, generando "ruido" en los datos.

Para reducir la distancia existente entre el modelo teórico y modelo práctico, se decide eliminar también aquellas secuencias que se repitan en el tiempo. Por ejemplo, si se reconoce

el evento atómico "*pushGasPedal*" durante 50 instancias, entonces quiere decir que se está realizando esa actividad de manera continuada en el tiempo. Sin embargo, la variable "tiempo" no está considerada dentro del modelo teórico definido, por lo que es interesante contemplar únicamente los eventos atómicos que indiquen un cambio de comportamiento del conductor, a fin de ser capaz de reconocer los diferentes eventos atómicos que componen una tarea. Incluir un número alto de secuencias repetidas (eventos atómicos repetidos en el tiempo) implica que las secuencias más relevantes de la tarea sean iguales, no pudiendo reconocer el resto de eventos atómicos que componen la tarea.

Para aplicar el trie, se parte una aplicación desarrollada por un miembro del grupo de investigación *CAOS* (Control, Aprendizaje y Optimización de Sistemas) de la Universidad Carlos III de Madrid (ver Figura 62 **¡Error! No se encuentra el origen de la referencia.**).

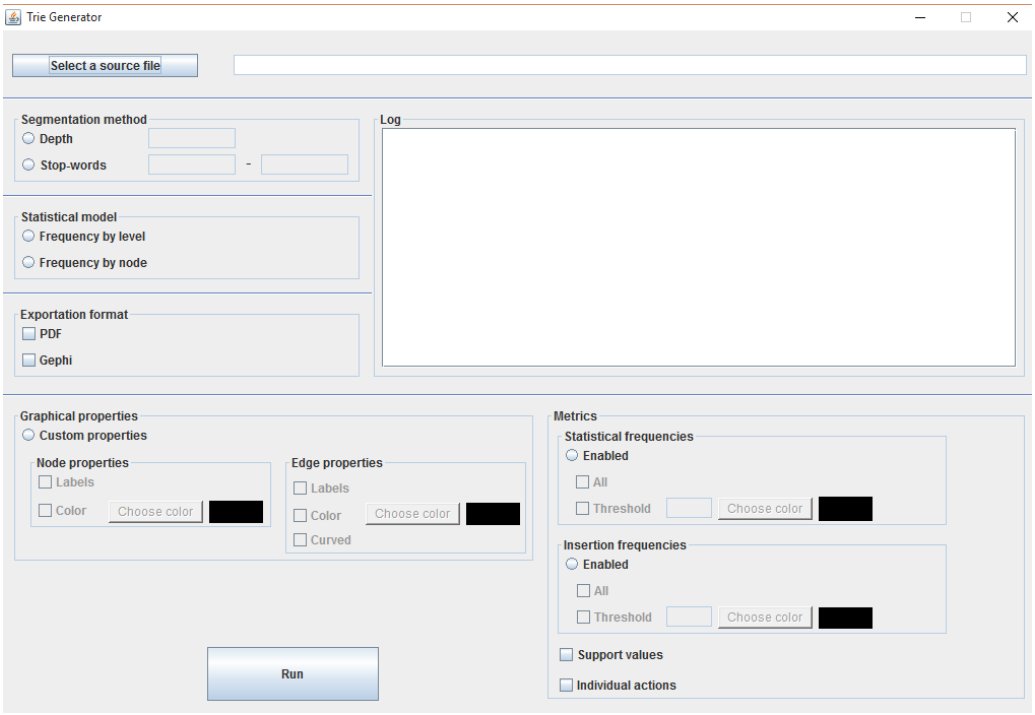

Figura 62: TrieGenerator

Como fichero de origen se establece el fichero de tarea "Detenerse" generado por el sistema, siendo necesario cambiar el carácter "-"por otro carácter (en este caso, por ":"). Esto se debe a que el programa utiliza el guión ("-") para separar secuencias y no permite leer dicho carácter.

Como valor de profundidad (longitud máxima de subsecuencas) se establece para esta primera aproximación, tras haber probado diferentes configuraciones, el valor de 5.

Dado que se desea extraer el valor de relevancia de cada subsecuencia (*support)*, se marca la opción de "Support Values". Tras ejecutar el programa, se generan todas las

subsecuencias con su *support* asociado. Las subsecuencias de mayor *support* son las siguientes:

Support	Subsecuencia
0.192	Root-cB_:RPM_pCP
0.190	Root-cB_:RPM
0.123	Root-cB_:RPM_pBP_pCP
0.115	Root-cB_:RPM_rBP_pCP
0.100	Root-cB_:RPM_pBP

Tabla 81: Support values detenerse

Como se observa en la tabla anterior, las secuencias más relevantes tienen como eventos atómicos "carBraking (cB)", ":RPM" (decreasingRevolutions), "pushBreakPedal (pBP)" y "pushClutchPedal (pCP)", siendo estos eventos concurrentes en todos los casos.

Al detener el coche, el coche va frenando, por lo que este primer modelo se aproxima con bastante acierto a lo ideal. Para reducir de marcha, el conductor debe de pisar el embrague constantemente (evento "pCP").

Es normal que no aparezca como secuencia relevante el evento de disminuir marcha ("changinGearDown"), ya que este evento sólo se produce 3 veces en todo el log original (sólo se reconoce dicho evento en el instante de cambiar una marcha superior a una inferior).

A pesar de que las secuencias más relevantes sean de profundidad 1, dichas secuencias están formadas por eventos concurrentes, por lo que cada secuencia agrupa un conjunto de eventos ocurridos en el mismo instante de tiempo.

De esta manera, se concluye que las secuencias con mayor support reflejadas en la tabla anterior son las secuencias que dominan la tarea, es decir, el modelo práctico futuro debe ser capaz de identificar la tarea en base a dichos eventos atómicos que componen las secuencias.

Se ha probado con diferentes configuraciones de profundidades, obteniendo modelos menos aproximados que éste, por lo que se decide mantener la profundidad 5.

5.4.3. Modelo de tareas: arrancar y salir

Una vez realizada la primera aproximación para la generación del modelo de tareas con "Detenerse", se decide realizar otro modelo de tareas a fin de comprobar que el enfoque dado con el uso de tries es correcto.

Se decide probar con otra sencilla tarea: "Arrancar y salir". En esta tarea, en el modelo ideal, se basa de los siguientes eventos:

- releaseGasPedal
- clutched
- changingGearUp

- pushGasPedal

Además, el vehículo debe pasar por los estados (eventos atómicos del vehículo) de "increasingRPM" y "carThrottle".

Al igual que en la tarea anterior, los eventos atómicos relacionados con la cara y ojos del conductor no se tienen en cuenta en el modelo teórico ideal, por lo que se deciden suprimir para no generar ruido en el modelo.

Asimismo, se eliminan aquellos eventos atómicos repetidos en el tiempo por el mismo motivo descrito en el apartado anterior.

Tras ejecutar el trie, las subsecuencias de mayor *support* son las siguientes:

Support	Subsecuencia
0.425	Root-cT_+RPM_pGP_pCP
0.375	Root-cT_+RPM_pGP_pCP- cT_+RPM_pGP_SW2R_pCP
0.375	Root-cT_+RPM_pGP_pCP- cT_+RPM_pGP_SW2R_pCP- cT_+RPM_pGP_pCP
0.35	Root-cT_+RPM_pGP_SW2R_pCP
0.343	Root-cT_+RPM_pGP_SW2R_pCP- cT_+RPM_pGP_pCP

Tabla 82: Support values arrancar y salir

Como se observa en la **¡Error! No se encuentra el origen de la referencia.**, las secuencias más relevantes tienen como subsecuencias (eventos atómicos) "cT (carThtootle)", "+RPM (increasingRPM)", "pGP (pushGasPedal)" y "clutched (pCP"), además de "SW2R (steering Wheel to right"). Si se comparan estas secuencias con las del modelo ideal, se observa un gran parecido, a excepción del evento "SW2R", el cual se puede considerar ruido del modelo.

En este caso, algunas secuencias relevantes son de nivel 2. Asimismo, cada secuencia agrupa diferentes eventos concurrentes, por lo que dentro de cada secuencia existen diferentes acciones que ocurren en el mismo instante de tiempo.

Tras haber descrito y analizado la primera aproximación del modelo de tareas para dos tareas, se concluye que la experimentación ha resultado ser bastante óptima, ya que los resultados obtenidos se parecen en gran medida a los esperados, siempre teniendo en cuenta que los modelos obtenidos son susceptibles de mejoras.

Como se ha comentado con anterioridad, este modelo práctico es una primera aproximación. Por ello, únicamente se han considerado dos tareas realizadas en dos escenarios diferentes.

Capítulo 6: Planificación y presupuestos

6. Planificación y presupuestos

En este apartado se especifica la planificación del presente proyecto, así como el presupuesto que costaría implementar este proyecto en una empresa, estimando los costes de personal y los costes de equipo (costes de hardware y costes de software).

6.1. Metodología de planificación

Para la planificación del proyecto, se ha decidido seguir un modelo en cascada (ver **¡Error! No se encuentra el origen de la referencia.**). Un modelo en cascada ordena las actividades de manera secuencial, de forma que el inicio de la siguiente actividad es el final de la actividad anterior. Se ha decidido utilizar esta metodología por su facilidad de entendimiento e implementación. Además, es una metodología orientada a documentos, por lo que es muy útil en el presente documento.

El modelo en cascada original, tal y como lo plantearon Winston W. Royce en 1970 y Barry Boehm en 1980, consta de las siguientes fases:

- **Análisis de requisitos**: en esta fase se analizan las necesidades del usuario final del software. Para ello, se deben de programar reuniones con el cliente (en este caso, el tutor del proyecto) para conocer exactamente que desea el cliente. El objetivo final de esta fase es tener un documento de requisitos que detalle claramente qué tiene que hacer el sistema a desarrollar y cómo tiene que hacerlo.

- **Diseño del sistema:** en esta fase, se divide el sistema en módulos o subsistemas claramente diferenciados, diseñando y especificando qué debe de hacer cada módulo en base a los requisitos obtenidos en la fase anterior. La salida de esta fase es el diseño de los distintos módulos del sistema, con el objetivo de su posterior implementación.

- **Implementación del sistema:** en base al diseño del sistema, se implementa el código fuente.

- **Pruebas del sistema:** una vez se ha implementado el sistema, se prueba para comprobar que el sistema funciona de manera adecuada y cumple con los requisitos definidos en la primera fase.

- **Verificación y mantenimiento:** una vez probado el sistema por el desarrollador, el usuario final (cliente) prueba el sistema y verifica su buen funcionamiento. Durante esta fase se mantiene el sistema en activo.

- **Documentación:** esta fase no está incluida dentro de las fases de un modelo clásico en cascada. Sin embargo, para este proyecto, esta fase se realiza una vez se ha probado y verificado el sistema.

A continuación se muestra un diagrama de la metodología en cascada:

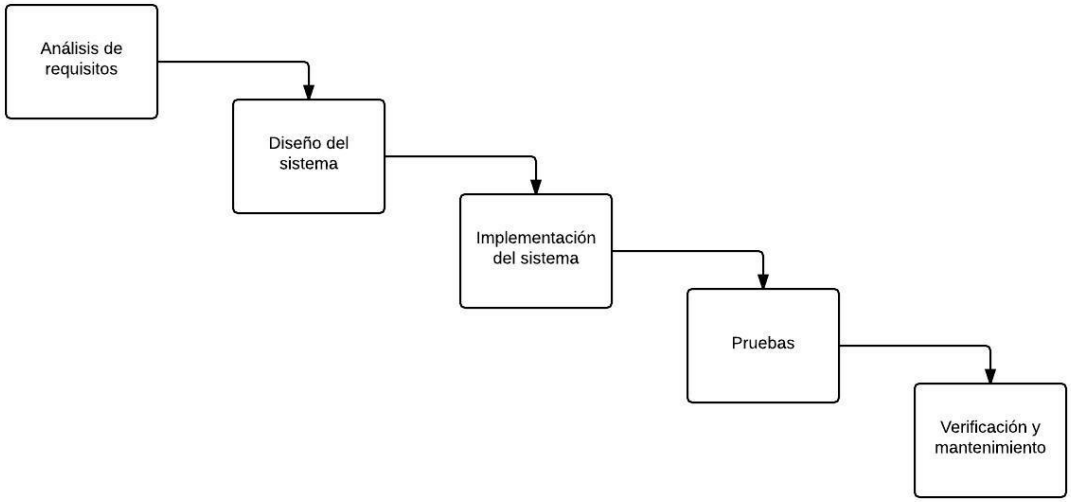

Figura 63: Planificación en cascada

6.2. Planificación

El presente proyecto comenzó el día 1 de Febrero de 2015. Está estimado y planificado un tiempo de duración de 7 meses, finalizando el mismo el 1 de Septiembre de 2015. A continuación, mediante una tabla se especifica la planificación del proyecto.

En la tabla se indica la duración de cada fase. Además, dentro de cada fase, se especifica la duración de cada tarea contenida en cada fase.

Tarea	Duración (días)	Fecha inicio	Fecha Fin
Análisis de requisitos	28	01/02/2015	20/02/2015
Reuniones iniciales con tutor	5	01/02/2015	05/02/2015
Especificaciones de requisitos	15	06/02/2015	20/02/2015
Casos de uso	8	21/02/2015	28/02/2015
Diseño del sistema	76	01/03/2015	15/05/2015
Arquitectura del sistema	10	01/03/2015	10/03/2015
Diseño de actividades	56	11/03/2015	05/05/2015
Diseño escenarios	10	06/05/2015	15/05/2015
Implementación sistema	60	16/05/2015	15/07/2015
Familiarización entorno STISIM	5	16/05/2015	20/05/2015
Estudio lenguaje C++	11	21/05/2015	31/05/2015
Discretización datos cámara	4	01/06/2015	04/06/2015

Procesado ficheros	10	05/06/2015	14/05/2015
Etiquetado	15	15/06/2015	30/06/2015
Modelo de tareas	15	01/07/2015	15/07/2015
Pruebas sistema	10	16/07/2015	25/07/2015
Verificación y documentación	37	26/07/2015	31/08/2015
Verificación por tutor	6	26/07/2015	31/07/2015
Documentación y presentación	31	01/08/2015	31/08/2015

Tabla 83: Planificación proyecto

A continuación se muestra un *diagrama de Gantt* (ver **¡Error! No se encuentra el origen de la referencia.**) que muestra la planificación del proyecto de manera gráfica:

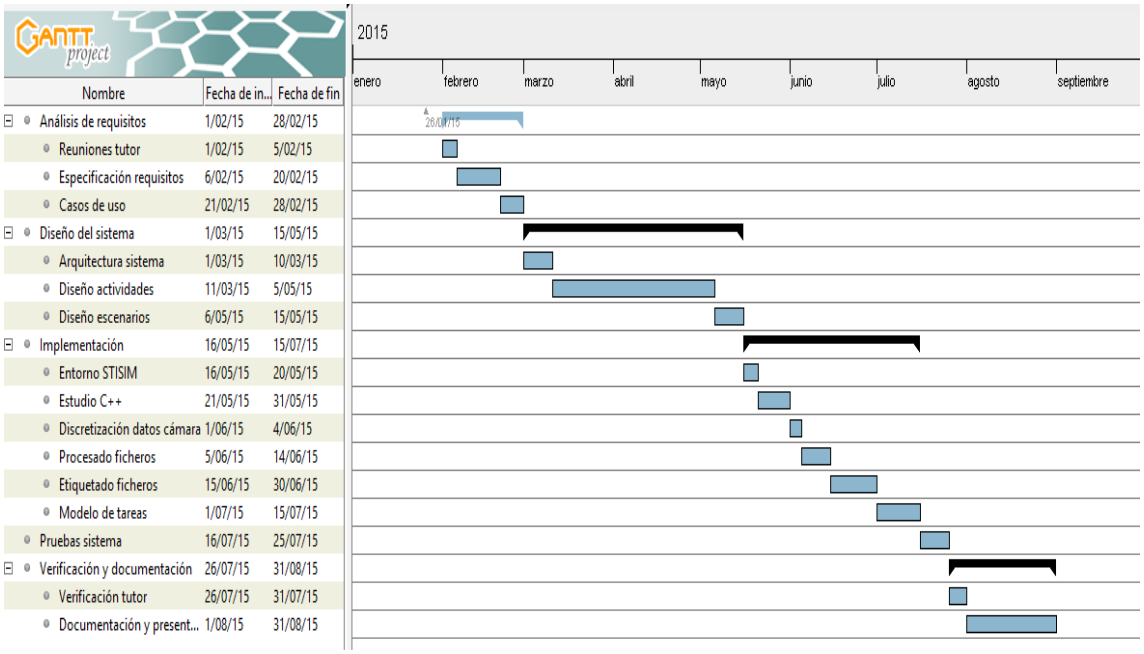

Figura 64: Planificación Gantt

6.3. Presupuesto

En este apartado se presenta el presupuesto del presente proyecto, identificando los roles de los participantes en él, estableciendo sueldos a cada persona por desempeñar su rol, estimando los costes materiales, ya sean hardware o software, y estimando los costes indirectos.

Los roles identificados en el presente proyecto son:

- **Jefe de proyecto**: es la persona encargada de dirigir y coordinar el presente proyecto.

- **Analista y diseñador:** persona encargada de la obtención de requisitos y del diseño del sistema.
- **Programador:** persona encargada de implementar el sistema.
- **Técnico de pruebas:** persona encargada de probar el sistema.
- **Investigador:** persona encargada de documentar el proyecto.

6.3.1. Costes de personal

En la siguiente tabla se muestran los roles identificados en el presente proyecto y el sueldo asignado (costes de personal). Destacar que los sueldos han sido establecidos a partir de la tabla salarial publicada en el BOE, nº 174 del 21 de julio de 2012, sección III, página 52601.

ROL	MESES TRABAJO	SALARIO MENSUAL	COSTE (€)
Jefe de proyecto	1,096	1.794,57 €	1966,84
Analista/diseñador	3,35	1.454,53 €	4872,675
Programador	1,93	1.129,90 €	2180,7
Técnico de pruebas	0,32	1.129,90	361,56
Investigador	1	669 €	669
TOTAL:			**10080,77 €**

Tabla 84: Costes de personal

6.3.2. Costes materiales

En la siguiente tabla se muestran los costes relacionados con los materiales, indicando el producto, el precio, el periodo de amortización del producto, el uso en meses y el coste para el presente proyecto, calculado como:

$$coste = \frac{precio}{periodo\ amortización} * uso$$

PRODUCTO	PRECIO (€)	PERIODO AMORTIZACIÓN (MESES)	USO (MESES)	COSTE PROYECTO (€)
Asus M70AD	900	48	7	134,04
STISIM M100	7500	48	7	1090,375
KINECT XBOX360	65	48	7	9,47
LICENCIA OFFICE 2013 PROFESSIONAL	539	60	7	62,93
LUCIDCHART	0	60	7	0,00
LICENCIA MICROSOFT VISUAL STUDIO	600	60	7	70
LICENCIA ECLIPSE	0	60	2	0,00
GANTT PROYECT	0	60	1	0,00
TOTAL:				**1385,81 €**

Tabla 85: Costes materiales

6.3.3. Costes totales

Tanto los costes materiales como los costes personales se denominan costes directos. A los costes directos es necesario añadirle el 10% del total de estos como costes indirectos, relativos al coste de Internet, luz, teléfono, etc.

TIPO COSTES	TOTAL
PERSONAL	10080,77 €
MATERIALES	1385,81 €
INDIRECTOS	1046,65 €
TOTAL:	**12513,23 €**

Tabla 86: Costes directos e indirectos

Además, se incluye un beneficio del 15% sobre el coste total:

TIPO COSTES	TOTAL
COSTE TOTAL	12513,23 €
BENEFICIOS (15%)	1876,95 €
TOTAL:	**14390, 18 €**

Tabla 87: Coste total proyecto

Por lo tanto, el presupuesto final del presente proyecto asciende a 14390,18 € (catorce mil trescientos noventa euros con dieciocho céntimos).

Capítulo 7: Conclusiones y Trabajos Futuros

7. Conclusiones y trabajos futuros

En este apartado se realizan las conclusiones, tanto técnicas como personales, así como las líneas futuras que se pueden seguir partiendo del presente proyecto.

7.1. Conclusiones técnicas

Tras haber realizado el presente proyecto, se concluye de manera técnica lo siguiente:

- Tanto el objetivo general como los objetivos específicos se han cumplido:
  - Se ha desarrollado de manera teórica un modelo ideal de actividades con el objetivo de aplicarlo a la práctica posteriormente. Este modelo consta de la definición teórica de los diferentes, eventos, acciones, tareas y maniobras que realiza un conductor de un vehículo.
  - Se ha conseguido implementar un sistema capaz de recoger datos de sensores y cámara, procesarlos, transformarlos y razonar con ellos para reconocer eventos atómicos en ellos.
  - Se ha conseguido desarrollar una primera aproximación del modelo de tareas, comparando los resultados con el modelo ideal de actividades, obteniéndose unos resultados bastante buenos.

- Tras haber realizado la experimentación, se ha comprobado que el paso del modelo teórico ideal a un modelo práctico capaz de reconocer actividades es una tarea compleja, en mayor medida cuando se desean reconocer actividades complejas. Entre los diversos factores que motivan esto se encuentran los eventos concurrentes y el tiempo, los cuales no se pueden definir en el modelo teórico.

- Tanto la extracción, como el procesado y la transformación de los datos, son unos de los aspectos más importantes del trabajo. Es importante estudiar qué variables son necesarias para cada escenario, así como las transformaciones que son necesarias aplicar a los datos para un mejor estudio de los datos. Un fichero de log de una tarea con variables no representativas de esa tarea es un fichero sin información útil, por lo que es fundamental elegir correctamente las variables.

7.2. Líneas futuras

En esta sección se estudian las líneas futuras que se pueden seguir a partir de este proyecto. Las más importantes se presentan a continuación:

- *Mejorar la aplicación de detección de ojos y cara*: la detección de los ojos y de la cara no se realiza todo lo bien que se desearía. En muchos instantes de tiempo, la aplicación encargada de ello no es capaz de detectar las posiciones y devuelve “no-detected”. Por otro lado, es necesario que las primeras imágenes que reciba la logre clasificar correctamente, ya que si no el resto de la ejecución fallará y devolverá datos

incorrectos. Por ello, una mejora sustancial de la aplicación implicaría que los ficheros de log tuvieran datos más precisos acerca de la posición de la cara y ojos.

- *Mejorar el modelo teórico de actividades:* pese a que se considera que el modelo teórico de actividades ideales es bastante completo, siempre es susceptible de mejoras. Una de estas mejoras consistiría en la definición de más eventos atómicos que sean capaces de formar parte de otras tareas, que a su vez sean capaces de generar otras maniobras.

- *Observar estados en el log:* existen estados en el modelo teórico, como "huecoAparcar=true" que no son observables directamente en el simulador y que requieren de una serie de cálculos intermedios para inferirlos. Por ello, se podría mejorar la implementación del sistema incluyendo estos cálculos que infieran aquellos estados no observables directamente en el simulador.

- *Generar un mayor número de logs:* en lugar de generar un único fichero de log para cada una de las tareas y maniobras definidas, se pueden generar varios ficheros por tarea/maniobra a fin de generar modelos de tareas/maniobras más precisos.

- *Generar un trie por tarea:* para cada tarea definida, generar un trie diferente tomando en cuenta el tiempo, es decir, manteniendo aquellos eventos atómicos que se repitan en el tiempo.

- *Utilizar diferentes tries:* es interesante discriminar la información que se desea representar en el trie. En el presente proyecto se obvia la información referente a los datos de la posición de cara y ojos del conductor. Como trabajo futuro, sería interesante generar para cada tarea diferentes tries: uno que contenga la información del conductor, otro que contenga la información del coche y otro que contenga la posición de cara y ojos del conductor.

- *Realizar un modelo de maniobras:* una vez generado el modelo de tareas mejorado, se podría generar un modelo de maniobras tal que al recibir una serie de tareas que forman una maniobra, el sistema sepa reconocer de qué maniobra se trata.

- *Eliminar el preprocesado manual:* para ello, sería necesario mejorar la codificación del sistema, generando un método capaz de eliminar toda la información superflua de los logs generados.

7.3. Conclusiones personales

En cuanto a las conclusiones personales, estoy muy satisfecho con la realización del presente proyecto, ya que se han cumplido todos los objetivos propuestos al inicio del mismo.

Nunca me había enfrentado a un proyecto de estas dimensiones, por lo que he aprendido lo difícil que es planificar y realizar un proyecto de gran complejidad.

Mi conocimiento del dominio del reconocimiento de actividades era nulo antes de la realización del proyecto, por lo que el inicio del proyecto fue bastante complejo, ya que tuve que leerme una gran cantidad de documentación para entender los aspectos básicos que me pudieran dar una base de conocimiento. Una vez me familiaricé con este área de estudio, el desarrollo teórico del modelo de actividades ha sido lo más costoso.

Además de adquirir conocimientos del área del reconocimiento de actividades, he aprendido otros conceptos de manera indirecta, como puede ser la programación en C++ o el desarrollo de unos presupuestos.

Por último, valor muy positivamente el interés del tutor mostrado en mí, solucionándome cada uno de los problemas que iban surgiendo y guiándome hasta conseguir terminar el proyecto con éxito.

Glosario de términos

- **Aprendizaje automático (*machine learning)***: rama de la Inteligencia Artificial que abarca diferentes técnicas que permiten dotar a los computadores de la capacidad de aprender modelos que permiten resolver problemas ya vistos.

- **C++:** lenguaje de programación orientado a objetos y con sintaxis del lenguaje C.

- **Conjunto de entrenamiento:** conjunto de datos en el que cada fila es una instancia y cada columna un atributo de dicha instancia. Este conjunto de datos se utiliza para que el computador "aprenda" un modelo.

- **Fichero log:** fichero de texto donde se escriben los eventos ocurridos en un sistema.

- **Java:** lenguaje de programación orientado a objetos, multiplataforma y fuertemente tipado, desarrollado por James Gosling.

- **Lógica borrosa:** rama de la Inteligencia Artificial que se basa en el razonamiento humano. La lógica borrosa asume que cada elemento tiene un grado de pertenencia a un conjunto.

- **Máquina de estados:** modelo matemático que representa los diferentes estados por los que puede pasar un determinado sistema. Dado un estado y una entrada, la máquina de estados transitará a otro estado, produciendo una salida.

- **Script:** pequeño programa que se ejecuta ante una serie de eventos.

- **STISIM:** software de simulación de conducción. Se utiliza principalmente para investigaciones.

Referencias

[1] D. A. Forsyth y J. Ponce, Computer Vision: A Moder Approach. 2nd edition, Prentice Hall Professional Technical Reference, 2011.

[2] R. Hernández García, E. García Reyes, J. Ramos Cózar y N. Guil Mata, «Modelos de representación de características para la clasificación de acciones humanas en vídeo,» *Revista Cubana de Ciencias Informáticas,* 2014.

[3] «https://fundacionmapfre.com,» [En línea].

[4] «http://www.dgt.es/es/prensa/notas-de-prensa/2015/20150105-mir_BALANCE-2014_seguridad-vial_nota.shtml,» [En línea].

[5] «http://www.stisimdrive.com/,» [En línea].

[6] G. Pelaez, «Head Pose Estimation Based on 2D and 3D for Driving Assistance Systems,» de *22nd WSCG International Conference on Computer Graphics, Visualization and Computer Vision*, Plzen, 2014.

[7] «http://www.xbox.com/,» [En línea].

[8] A. Ledezma Espino, «Reconocimiento de Actividades: anticipando las necesidades del usuario».

[9] D. Weilan, R. Ronfard y E. Boyer, «A Survey Of Vision-Based Methods for Action Representation, Segmentation and Recognition,» 2011.

[10] T. Lan, L. Chen, Z. Deng, Z. Guang-Tong y G. Mori, «Learning Action Primitives for Multilevel Video Event Understanding».

[11] G. Johansson, «Visual perception of biological motion and a model forits analysis, perception & psychophysics,» 1973.

[12] T. Darrell y A. Pentland, «Space-time gestures,» de *Conference on Computer Vision and Pattern Recognition*, 1993.

[13] L. Fei-Fei, R. Fergus y A. Torralba, «Bag of Features Models».

[14] K. Torkkola, M. Gardner, C. Schreiner, K. Zhang, B. Leivian, K. Zhang y J. Summers, «Understanding Driving Activity Using Ensemble Methods,» 2008.

[15] «http://es.wikipedia.org/wiki/Gram%C3%A1tica_libre_de_contexto_probabil%C3%ADstic

] a,» [En línea].

[16 J. K. Aggarwal y M. S. Ryoo, «Human Activity Analysis: A Review,,» ACM COMPUTING,
] 2011.

[17 «https://en.wikipedia.org/wiki/Advanced_driver_assistance_systems,» [En línea].
]

[18 *SDDL_Events_Manual. Manual del simulador STISIM.*
]

[19 «https://es.wikipedia.org/wiki/Top-down_y_bottom-up,» [En línea].
]

[20 «http://www.boe.es/buscar/act.php?id=BOE-A-1999-23750,» [En línea].
]

[21 J. Antonio Iglesias, A. Ledezma Espino y A. Sanchís, «An Evolving Framework for Clustering
] Computer Users».

[22 M. Ryoo, I. Laptev, G. Mori y S. Oh, «Emerging Topics in Human Activity Recognition. CVPR
] Tutorial,» 2014.

[23 S. O'Hara y B. Draper, «Introduction to the bag of features paradigm for image
] classification and retrieval,» 2010.

[24 «http://www.motorafondo.net/aparcamiento-automatico-para-el-futuro/,» [En línea].
]

[25 «http://www.autofacil.es/novedades/2012/07/06/coches-aviso-cambio-carril-
] accidentes/10385.html,» [En línea].

[26 «http://www.circulaseguro.com/que-es-la-deteccion-de-las-senales-de-trafico/,» [En
] línea].

ANEXO A: MANUAL DE USUARIO

En este anexo se describe un manual de usuario para poder generar ficheros de datos etiquetados a partir de la realización de una actividad en el simulador. A continuación se incluyen los diferentes pasos a seguir para realizar esta tarea:

1. Lo primero que se debe hacer es instalar el software del simulador STISIM en el equipo deseado. Una vez instalado, se debe ejecutar el software clicando sobre el siguiente icono:

Figura 65: icono STISIM Drive 3

2. Seguidamente, se debe de encender el equipo que contiene instalada la aplicación de detección de ojos y cara. Una vez encendido, se ejecutará la aplicación de detección de ojos y cara (*head_eyes_estimation)*.

3. Una vez ejecutado el simulador, se debe de elegir qué actividad se quiere realizar y en qué escenario se quiere realizar. Una vez decidido, se debe de editar el escenario correspondiente para incluir una sentencia en el fichero del escenario que permita extraer el valor de las variables deseadas. Para más información, ver **¡Error! No se encuentra el origen de la referencia..**

4. Una vez decidido la actividad a realizar y el escenario donde se realizará, se debe configurar la pantalla inicial del simulador (*ver figura 63)*.
   4.1. En el primer desplegable, denominado *Configuration Files*, se debe de poner el fichero *STISIM3_OM_ManualTransmission.Cfg.*
   4.2. En el segundo desplegable, denominado *Scenario Files*, se debe de elegir el escenario elegido y editado en el paso 3. Los escenarios se encuentran, por defecto, en *C:\STISIM\PDeS*.
   4.3. En el tercer desplegable, se debe de elegir el nombre del fichero de salida (fichero log) que generará el simulador una vez finalice la ejecución de la actividad. Este fichero se genera, por defecto, en la carpeta de instalación del simulador.

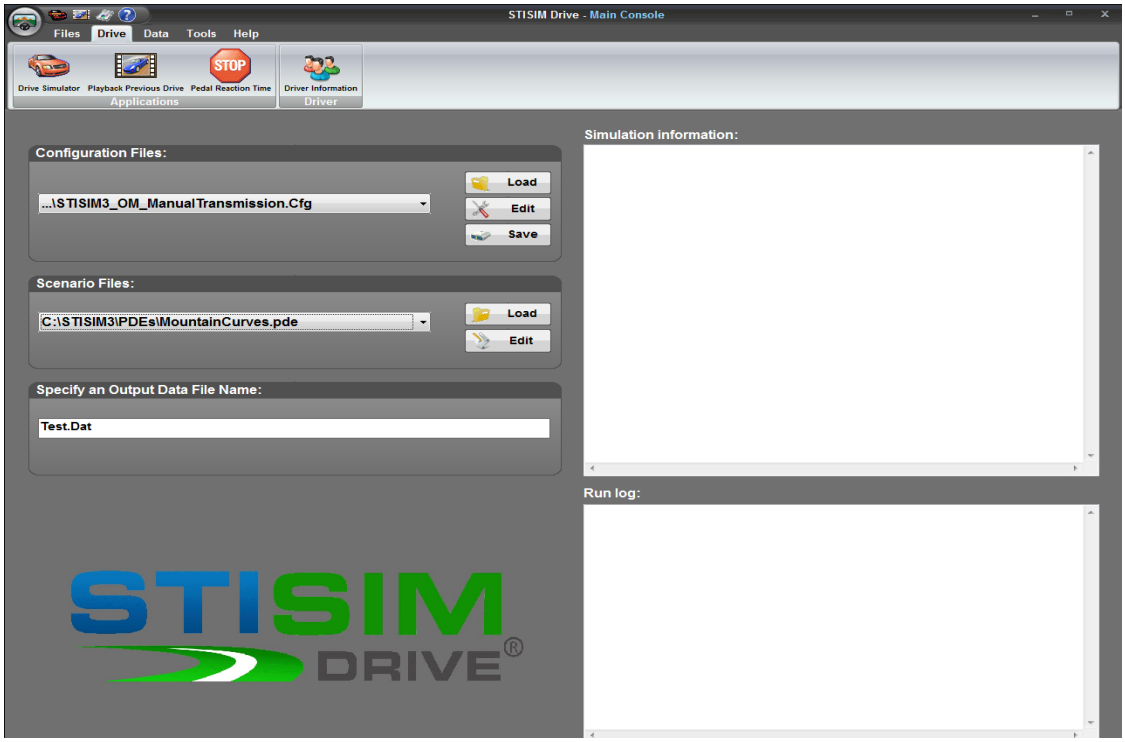

Figura 66: configuracion simulador

5. Una vez establecida la configuración, se debe de realizar la actividad en el simulador. Para ello, es necesario clicar sobre *Drive Simulador* (esquina superior izquierda).
6. Tras acabar la actividad se generan dos ficheros:
   6.1. Fichero *Test.Dat*: contiene el valor de las variables que se extraen del simulador para cada instante de tiempo definido (***ver ANEXO B: ESCENARIOS)***. Además, contiene otra información innecesaria (cabeceras y otro tipo de información). A continuación se indica el preprocesado manual que hay que realizar al fichero:

CABECERA

La cabecera del fichero es innecesaria para el procesado de los datos, por lo que se debe de eliminar manualmente.

```
Date: June 19, 2015
Time: 1:50:19:62 PM
ID:
Test Computer: PC-SIMULADOR
Scenario File: C:\STISIM3\Scenarios (English Units)\Driving on Left\Signal_Light_Practice-PASOCEBRA.Evt
Configuration File: E:\Simulador\Configuraciones\STISIM3_OM_ManualTransmission.Cfg
License Serial Number: 74757

Name:
Run Number:
Comments:
```

Figura 67: cabecera fichero dat

DEFINICIÓN DEL ESCENARIO

La definición del escenario es innecesaria también, por lo que debe de eliminarse por completo.

```
Scenario:

Config_Start_Y = 6{2}
Config_Speed0m_Units = KPH
Config_Output_Units = METRIC
Config_Road_Side = Left
0, Roadway, 10, 2, 1, 3, 0.5, 10, 10, 0.4, 0.4, 0, -1, C:\STISIM3\Data\Textures\Road15.Jpg, 255/255/255, 12, 0, -1, C:\STISIM3\Data\Textures\
0, Speed Limit, 48,0
0, Static Object,100,-62.3359603881836{0},0,0,0,0, C:\STISIM3\Data\EuroSigns\Speed Limit\Speed_Limit_70.Mka,
0, Intersection, 0,120, 1, 4, 0, 0
0, Pedestrian#1,120, 10, 0.9144, 1.524{1}, R, *6~8
10, Begin Block Save, 1, .0333, PRUEBA, 1, 23, 27, 28, 52, 25, 51, 26, 7, 46#1
67.140000174408, Roadway, 10, 2, 1, 3, 0.5, 10, 10, 0.4, 0.4,19.8119998488464,0,C:\STISIM3\Data\Textures\Road15.Jpg,255/255/255,12,0,0,C:\STI
172.859999825592, Roadway, 10, 2, 1, 3, 0.5, 10, 10, 0.4, 0.4,19.8119998488464, -1, C:\STISIM3\Data\Textures\Road15.Jpg, 255/255/255, 12, 0,
738, Static Object,762,-62.3359603881836{0},0,0,0,0, C:\STISIM3\Data\EuroSigns\Speed Limit\Speed_Limit_70.Mka,
1800, Vehicles, 1000, -1000{0}, 0{1}, 4, S43, 1, 1
2038, Intersection, 0,762, 1, 0, 0, 0
2238, Static Object,762,-62.3359603881836{0},0,0,0,0, C:\STISIM3\Data\EuroSigns\Speed Limit\Speed_Limit_70.Mka,
2238, Building,762, -30{0}, B2
2556, Vehicles, 1000, -800{0}, 50{0}, 4, S43, 1, 1
```

Figura 68: escenario fichero dat

DATOS

Los datos dinámicos generados son los que realmente interesan. Se debe de eliminar las dos primeras líneas (“Dynamic blocks” y “Block X: nombre”).

```
Dynamic blocks:

Block #1:   PRUEBA,829
```

3.600	32.28	0	65535	56953	1	1433.543	32751	-4.01	109.87	11.52	0.00
3.633	32.76	0	65535	56953	1	1434.791	32751	-4.01	109.57	11.52	0.00
3.667	33.24	0	65535	56953	1	1436.037	32751	-4.01	109.26	11.52	0.00
3.700	33.72	0	65535	56953	1	1437.283	32751	-4.01	108.94	11.52	0.00
3.733	34.21	0	65535	56953	1	1438.527	32751	-4.01	108.62	11.52	0.00
3.767	34.69	0	65535	56953	1	1439.771	32751	-4.01	108.30	11.52	0.00
3.800	35.17	0	65535	56953	1	1441.013	32751	-4.01	107.97	11.52	0.00
3.833	35.65	0	65535	56953	1	1442.254	32751	-4.01	107.64	11.52	0.00
3.867	36.12	0	65535	56953	1	1443.493	32751	-4.01	107.30	11.52	0.00
3.900	36.60	0	65535	56953	1	1444.732	32751	-4.01	106.96	11.52	0.00
3.933	37.08	0	65535	56953	1	1445.969	32751	-4.01	106.62	11.52	0.00
3.967	37.56	0	65535	56953	1	1447.205	32751	-4.01	106.27	11.52	0.00
4.000	38.04	0	65535	56953	1	1448.439	32751	-4.02	105.91	11.52	0.00
4.033	38.51	0	65535	56953	1	1449.673	32751	-4.02	105.55	11.49	0.91
4.067	38.99	0	65535	56953	1	1450.905	32751	-4.02	105.19	11.46	0.91
4.100	39.46	0	65535	56953	1	1452.136	32751	-4.02	104.82	11.43	0.91
4.133	39.94	0	65535	57213	1	1453.366	32751	-4.02	104.45	11.40	0.91
4.167	40.41	0	65535	57213	1	1454.595	32751	-4.02	104.08	11.37	0.91
4.200	40.89	0	65535	57213	1	1455.822	32751	-4.02	103.69	11.34	0.91
4.233	41.36	0	65535	57213	1	1457.048	32751	-4.02	103.31	11.31	0.91
4.267	41.83	0	65535	57213	1	1458.273	32751	-4.02	102.92	11.28	0.91
4.300	42.31	0	65535	57213	1	1459.496	32751	-4.02	102.53	11.25	0.91
4.333	42.78	0	65535	57213	1	1460.718	32751	-4.02	102.13	11.22	0.91
4.367	43.25	1279	65535	57213	1	1455.108	32751	-4.02	101.72	11.19	0.91
4.400	43.72	1279	65535	57213	1	1456.328	32739	-4.02	101.32	11.16	0.91

Figura 69: datos fichero dat

PIE DE FICHERO

Contiene un resumen de la conducción del usuario. Por lo tanto, es información innecesaria y se debe de eliminar.

```
0 0 0 0 0 0 0

Intersection turns (Number, Expected turn, Actual turn, Correctness of turn, Vehicle #, Gap distance, Vehicle #, Gap time):

  1  0  0  0

Turn signal usage (Total, Turns, Lane changes):

Good performance =      0     0     0
Poor performance =      0     0     0
Missed =                0     0     0

Internally triggered events (Event, Action, Time, Distance):

Pedestrian1                        0      4.03      14.45

Driver mistakes:

Total number of off road accidents = 0
Total number of collisions = 0
Total number of pedestrians hit = 0
Total number of speed exceedances = 0
Total number of speeding tickets = 0
Total number of traffic light tickets = 0
Total number of stop signs missed = 0
Total number of centerline crossings = 0
Total number of road edge excursions = 0
Total number of stops at traffic lights = 0
Total number of correct DA responses = 0
Total number of incorrect DA responses = 0
Total number of DAs with no response = 0
Total run length (Drive T, X, Total T) = 31.20             130          31.20
Total number of illegal turns =              0
Total number of low speed warnings =         0
Total number of high speed warnings =        0
Over speed limit (% Time, % Distance) = 4.91   16.52
Out of lane (% Time, % Distance) = 0.00   0.00

Individual mistakes (Time, Distance, Elapsed distance or object number, Elapsed time, Maximum value, Speed limit/Centerline):

Speed exceedance           6.00       39.00       17.25        1.23    14.49    13.33
Speed exceedance          30.90      125.41        4.15        0.30    14.06    13.33
```

Figura 70: pie de fichero dat

Por lo tanto, el fichero generado, debe de contener únicamente los datos del apartado "Datos" (*ver figura 68)*, siendo la primera fila del fichero la primera instancia de datos. Esto es importante porque el programa de procesado asume que la primera línea del fichero es ya una instancia.

```
20.600   90.36   65535   3583   26111   0   2381.604   32751   2.68   99.62   -12.19   0.00   110.60   -12.19   0.00
20.633   89.47   65535   3327   18943   0   2300.357   32751   2.68   98.80   -12.19   0.00   109.77   -12.19   0.00
20.667   88.59   65535   3583   14847   0   2223.888   32751   2.68   97.98   -12.19   0.00   108.96   -12.19   0.00
20.700   87.71   65535   3583    9983   0   2151.917   32751   2.69   97.18   -12.19   0.00   108.15   -12.19   0.00
20.733   86.83   65535   3583    5887   0   2084.177   32751   2.69   96.38   -12.19   0.00   107.35   -12.19   0.00
20.767   85.94   65535   3583    2815   0   2020.421   32751   2.69   95.58   -12.19   0.00   106.56   -12.19   0.00
20.800   85.06   65535   4095       0   0   1960.414   32751   2.69   94.80   -12.19   0.00   105.77   -12.19   0.00
20.833   84.18   65535   5375       0   0   1903.936   32751   2.69   94.03   -12.19   0.00   105.00   -12.19   0.00
20.867   83.30   65535   5887       0   0   1850.779   32751   2.69   93.26   -12.19   0.00   104.23   -12.19   0.00
20.900   82.41   65535   6911       0   0   1800.748   32751   2.69   92.50   -12.19   0.00   103.47   -12.19   0.00
20.933   81.53   65535   7935       0   0   1753.659   32751   2.69   91.75   -12.19   0.00   102.72   -12.19   0.00
20.967   80.65   65535   8191       0   0   1709.339   32751   2.70   91.01   -12.19   0.00   101.98   -12.19   0.00
21.000   79.79   65535   8959       0   0   1667.626   32751   2.70   90.27   -12.19   0.00   101.24   -12.19   0.00
21.033   78.95   65535   10495      0   0   1628.365   32751   2.70   89.54   -12.19   0.00   100.52   -12.19   0.00
21.067   78.13   65535   11519      0   0   1591.414   32763   2.70   88.83   -12.19   0.00    99.80   -12.19   0.00
21.100   77.33   65535   12799      0   0   1556.635   32763   2.70   88.11   -12.19   0.00    99.09   -12.19   0.00
21.133   76.60   65535   17151      0   0   1523.901   32763   2.70   87.41   -12.19   0.00    98.38   -12.19   0.00
21.167   75.94   65535   21247      0   0   1493.093   32763   2.70   86.71   -12.19   0.00    97.68   -12.19   0.00
21.200   75.32   65535   25343      0   0   1464.096   32763   2.70   86.01   -12.19   0.00    96.99   -12.19   0.00
21.233   74.78   65535   30207      0   0   1436.804   32763   2.71   85.32   -12.19   0.00    96.30   -12.19   0.00
21.267   74.24   65535   31487      0   0   1411.118   32763   2.71   84.64   -12.19   0.00    95.61   -12.19   0.00
21.300   73.68   65535   31231      0   0   1386.941   32776   2.71   83.96   -12.19   0.00    94.93   -12.19   0.00
21.333   73.11   65535   30719      0   0   1364.187   32788   2.71   83.28   -12.19   0.00    94.26   -12.19   0.00
21.367   72.55   65535   30975   2303   0   1342.771   32788   2.71   82.62   -12.19   0.00    93.59   -12.19   0.00
21.400   71.99   65535   31231   7935   0   1322.614   32816   2.71   81.95   -12.19   0.00    92.92   -12.19   0.00
21.433   71.44   65535   31487   15615  0   1303.642   32844   2.71   81.29   -12.19   0.00    92.27   -12.19   0.00
21.467   70.89   65535   31487   21759  0   1285.786   32900   2.71   80.64   -12.19   0.00    91.61   -12.19   0.00
```

Figura 71: fichero ejemplo preprocesado manualmente

6.2. Fichero *Data.csv*: contiene el valor de la posición de la cara y los ojos, aparte de otra información innecesaria. El fichero contiene una primera línea de cabecera que debe de eliminarse manualmente. El fichero, por lo tanto, debe de tener el siguiente estilo (*ver figura 69*).

```
28.29991;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.33324;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.36658;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.39991;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.43324;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.46658;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.49991;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.53324;0;1;1;0;0;104.73;0;4.04;up-left;No-Detected
28.56657;0;1;1;0;0;104.72;0;4.04;up-left;No-Detected
```

Figura 72: fichero cara y ojos

7. Una vez se tienen los dos ficheros limpiados manualmente, se procede al procesamiento, transformación y etiquetado de los ficheros. Para ello, se debe de ejecutar el programa, estableciendo en el método "*Main*" los parámetros correctos en las llamadas a los métodos que realizan estas tres tareas.

7.1. En el método *combineFiles*, se pasan como parámetros las rutas de los ficheros del simulador (.dat y .csv) y la ruta del fichero a generar como combinación de ambos ficheros:

```
file.combineFiles("C:\\Users\\Alvaro\\Desktop\\logs\\limpieza_manual\\adelantar.Dat",
"C:\\Users\\Alvaro\\Desktop\\logs\\originales\\adelantar.csv",
"C:\\Users\\Alvaro\\Desktop\\logs\\procesados\\adelantar_process.csv"
);
```

Figura 73: método combineFfiles

7.2. En el método *saveInMemory* se pasa como parámetro el fichero generado en el apartado anterior (fichero *process,* es decir, el tercer parámetro del método anterior). Este método guarda en memoria los datos.

```
file.saveInMemory("C:\\Users\\Alvaro\\Desktop\\logs\\procesados\\adelantar_process.csv");
```

Figura 74: método saveInMemory

7.3. En el método *labelAtomic* se pasan como parámetros el array de datos guardados en el apartado anterior, la ruta del fichero a generar y la cabecera del fichero (debe de ser generada a mano).

```
at.labelAtomic(Log.datos,"C:\\Users\\Alvaro\\Desktop\\logs\\atomicos\\adelantar.csv",
"C:\\Users\\Alvaro\\Desktop\\logs\\cabeceras\\cabecera_adelantar.txt"
);
```

Figura 75: método labelAtomic

7.4. Finalmente, ya se tiene un fichero con los datos normalizados, transformados y etiquetados con eventos atómicos (uno o varios eventos atómicos por cada instancia). A continuación, en función de si la actividad es una maniobra o una tarea, se ejecuta el método *labelTask o labelManeuver.* Dicho método crea un fichero separado por comas. Entre cada coma se representa/n el/los evento/s atómico/s de cada instancia. Finalmente, se escribe el nombre de la maniobra/tarea. Este último paso se realiza para crear el modelo de tarea/maniobra, es decir, el conjunto de eventos atómicos realizados en la tarea/maniobra representa a dicha tarea/maniobra.

7.5. Se aplica un trie mediante la aplicación *TrieGenerator* sobre el fichero de tareas para obtener la secuencia de eventos más representativos.

ANEXO B: ESCENARIOS

En este anexo se describen los escenarios utilizados en la realización de las actividades ideales del modelo jerárquico de ideales. Primeramente se define como extraer las variables del simulador. Posteriormente, se describen los escenarios elegidos para la realización de las actividades ideales.

B.1. Extracción de variables del simulador

Para poder extraer las variables del simulador, es necesario incluir una sentencia en el escenario donde se realiza la actividad. Un ejemplo de la línea a incluir es la siguiente:

```
30, Begin Block Save, 1, .0333, PRUEBA, 1, 23, 27, 28, 52, 25, 51, 26, 7, 46#1
```

- El primer parámetro representa la distancia en metros a la que se lanza el evento de empezar a guardar datos. En este caso, se empezarán a generar los datos cuando el conductor haya recorrido 30 metros.
- El segundo parámetro indica que se deben de guardar los datos (*Begin Block Save)*.
- El tercer parámetro puede valor 1 o 0. Si vale 1, entonces las instancias se generan cada “x” segundos, siendo “x” el cuarto parámetro (en este caso y en todos, dado los requisitos, es cada 0.033 segundos). Si vale 0, entonces las instancias se generan cada “x” metros.
- El quinto parámetro representa el nombre que se le da al bloque de generación de datos (simplemente representativo).
- Los “n” siguientes parámetros representan las variables que se desean extraer. Para conocer qué variables están disponibles, consultar los manuales oficiales del simulador.

B.2. Escenarios elegidos

ARRANCAR Y SALIR

El escenario elegido para la realización de esta tarea es: *Stopping_Practice.Evt*. Este escenario es un escenario básico que permite al usuario practicar el avanzar y el detenerse. Consta de una carretera básica, algunos vehículos y algunas señales.

Se indica en el escenario que se desean extraer cada 0.033 segundos las siguientes variables: *tiempo (s), velocidad (km/h), pedal acelerador, pedal freno, pedal embrague, marcha actual, rpm y ángulo del volante*. Tal y como se describe en el apartado *4.1.1. DATOS SIMULADOR*, estos son las variables básicas que son comunes a todas las tareas y maniobras, es decir, estas variables siempre se van a extraer ya que se consideran básicas para el posterior etiquetado de actividades. El rango de estas variables ha sido descrito en la tabla 11.

Todos los datos de todas las tareas/maniobras se extraen cada 0.033 segundos, generando instancias de información, por lo que esta información será omitida en la descripción del resto de tareas/maniobras.

DETENER COCHE

El escenario elegido para la realización de esta tarea es: *Mountain_Curves.pde,* un escenario con curvas en montaña. Las variables que se extraen para esta tarea son las descritas en la tabla 11 del presente documento, es decir, las denominadas "variables básicas". Es importante destacar que en esta tarea se ha decidido empezar a generar datos una vez hayan transcurrido 300 metros, ya que para esta tarea no interesan los datos generados cuando el vehículo está detenido inicialmente y cuando está acelerando, sólo interesan los datos generados desde que el vehículo comienza a frenar.

MARCHA ATRÁS

El escenario elegido para la realización de esta tarea es: *Stopping_Practice.Evt*. Las variables que se extraen para esta tarea son las descritas en la tabla 11 del presente documento, es decir, las denominadas "variables básicas". Es importante destacar que en esta tarea se ha decidido empezar a generar datos una vez han transcurrido 150 metros, ya que el sistema no permite dar marcha atrás desde el principio, por lo que se mueve el vehículo 150 metros hacia delante y se detiene, empezando a generar datos una vez el vehículo se encuentra en el estado inicial.

CAMBIAR CARRIL DERECHA/IZQUIERDA

El escenario elegido para esta tarea es *Right_Turns_Easy.evt*. Dicho escenario consta de una carretera con cuatro carriles, dos en cada sentido separados por una mediana.

Las variables que se extraen en esta tarea son las básicas definidas en la tabla 11 y, además, la distancia del vehículo a la mediana. Se considera que esta variable es fundamental para el reconocimiento de esta tarea. Esta variable toma un valor 0 en la mediana y más negativo cuanto más se aleja el vehículo de la mediana. Es decir, cuando el vehículo se

encuentre en el carril izquierdo, esta variable tomará un valor muy negativo (en torno a -18). Cuando el vehículo se encuentre en el carril derecho, esta variable tomará un valor poco negativo (en torno a -6).

Un cambio de carril del izquierdo al derecho, por ejemplo, se reflejará en la variable "*distanciaMediana*", ya que el valor irá siendo cada vez más cercano a 0.

Dado que el vehículo siempre empieza en el carril izquierdo, para la tarea de "Cambio carril izquierda", se generan los datos cuando el vehículo ha transcurrido 150 metros. El objetivo de esto es que el vehículo se sitúe en el estado inicial (en este caso, en el carril derecho).

ADELANTAR

El escenario seleccionado para la realización de esta maniobra es *Behind_VehiclePass.pde*. Este escenario se ha modificado y únicamente consta de una carretera de dos carriles y de un vehículo amarillo que se encuentra delante del conductor. El vehículo delantero acelera desde las 20mph hasta las 60mph.

Las variables seleccionadas a extraer cada 0.033 segundos son las variables básicas definidas en la tabla 11 y, además, los datos del vehículo a adelantar. El software del simulador permite extraer datos del vehículo seleccionado. Sin embargo, únicamente permite extraer las siguientes variables:

- Distancia del vehículo al vehículo del conductor
- Distancia del vehículo a la mediana
- Velocidad del vehículo

Además, al indicar que se desean extraer los datos de un vehículo, se generan automáticamente estas tres variables, sin posibilidad de filtrar la que no se desee. Por ello, los valores de estas tres variables formarán parte del fichero log generado. De estas tres, tanto la distancia del vehículo del conductor al vehículo a adelantar, como su velocidad, son muy interesantes, ya que permiten saber si el vehículo está suficientemente cerca como para adelantarle y si su velocidad es inferior a la del conductor.

CEDER PASO PEATONES

El escenario seleccionado para la realización de esta maniobra es *Signal_Light_Practice.evt.* Este escenario ha sido modificado, eliminando el semáforo contenido en él y añadiendo en su lugar un paso de peatones con un peatón cruzando.

Las variables elegidas a extraer son las variables básicas (ver tabla 11), la distancia del vehículo a la mediana (para conocer en qué carril se encuentra) y los datos del peatón. Al igual que sucede con los vehículos, el simulador permite conocer tres variables de los peatones:

- Distancia del conductor al peatón
- Distancia del peatón a la mediana
- Velocidad del peatón

Estas tres variables son sumamente importantes. Cuanto menor sea la distancia del conductor al peatón, más cerca estará el conductor del paso de cebra y más tendrá que frenar. Por otro lado, la distancia del peatón a la mediana aumenta o disminuye según el peatón se encuentre cruzando hacia un lado u otro de la carretera. Por último, la velocidad del peatón indica como de rápido va el mismo y, por lo tanto, indica al conductor si debe de detenerse o, por el contrario, el peatón cruzará el paso de cebra antes de que el conductor llegue al mismo.

MANTENER DISTANCIA SEGURIDAD

El escenario elegido para la realización de esta maniobra es *Behind_Vehicle_Pass.pde*. Este escenario se ha modificado y únicamente existe una carretera y un vehículo que se encuentra delante del vehículo del conductor.

Las variables elegidas son las básicas (ver tabla 11), además de la distancia del vehículo a la mediana y los datos del vehículo delantero. Estos datos son exactamente los mismos que los de la maniobra *Ceder Paso de Peatones*. Tanto la velocidad del vehículo delantero como la distancia de ese vehículo al del conductor son sumamente importantes para esta maniobra, por lo que se considera fundamental su presencia para un futuro reconocimiento de esta maniobra.

DETENERSE ANTE SEMÁFORO

El escenario elegido para esta maniobra es *Signal_Light_Practice.Evt*. Se ha modificado para que aparezca un semáforo en rojo a los 200 metros de haber iniciado la marcha.

Las variables seleccionadas a extraer en cada golpe de tiempo son las básicas (ver tabla 11), el estado del semáforo y la distancia recorrida.

El estado del semáforo es fundamental para saber si está en rojo (devuelve un 3), en verde (devuelve un 1) o en ámbar (devuelve un 2).

Por otro lado, no existe ninguna variable que indique la distancia del vehículo al semáforo. Sin embargo, se puede saber la distancia recorrida por el vehículo y la posición del semáforo. Con estos datos, se infiere la distancia del vehículo al semáforo restando la posición del semáforo con la distancia recorrida.

APARCAR EN LÍNE A IZQUIERDAS/DERECHAS

El escenario elegido para estas maniobras ha sido *Town_Square.pde*. Este escenario es un entorno de ciudad, con vehículos aparcados a ambos lados de la carretera, por lo que es óptimo para la realización de estas maniobras. Se sitúan, mediante el lenguaje de script del simulador, dos vehículos separados una distancia de unos 8 metros a ambos lados de la carretera, es decir, cuatro vehículos. De esta manera, existe un hueco a la izquierda del conductor y otro a la derecha, con el objetivo de poder realizar cualquiera de las dos manioras.

Las variables elegidas son las básicas (ver tabla 11), la distancia a la mediana y los datos de ambos vehículos (en caso de aparcar a izquierdas, los datos de los dos vehículos estacionados a la izquierda del conductor). Se recuerda que estos datos son la velocidad (en

este caso, cero), la distancia a la mediana y la distancia del vehículo al vehículo del conductor (muy importante para medir distancias).

APARCAR BATERÍA FRENTE DERECHA/IZQUIERDA

El escenario elegido para la realización de estas maniobras es *ParkingLot_Demo.evt*, el cual consta de un parking al final de una carretera, por lo que es óptimo para estas maniobras.

Las variables elegidas son las básicas (ver tabla 11) y los datos de los vehículos a los lados del hueco para aparcar. Estos datos, como se ha comentado en repetidas ocasiones, son la velocidad de los vehículos, la distancia a la mediana y la distancia de los vehículos al vehículo del conductor. Esto último es importante, ya que permite conocer si existe hueco para aparcar.

ANEXO C: REGLAS DE EVENTOS ATÓMICOS

Para etiquetar las instancias con eventos atómicos se produce un razonamiento con los datos obtenidos del simulador y de la cámara, razonando entre la instancia del instante "t" y la instancia del instante "t+1" (excepto la última instancia, que se define en base a ella misma).

Algunas variable irá acompaña de un número, siendo "1" la instancia en el instante "t" y "2" la instancia en el instante "t+1".

Las reglas para etiquetar los eventos atómicos son las siguientes:

Condición	Evento atómico	Descripción	Comentarios
Speed1 < Speed2	cT	carThrootle	Coche acelera
Speed1 > Speed2	cB	carBraking	Coche frena
Rpm1 < Rpm2	+RPM	increasingRevolutions	Suben rpm
Rpm1 > Rpm2	-RPM	decreasingRevolutions	Bajan rpm
gasPedal1 < gasPedal2	pGP	pushGasPedal	Pisar acelerador
gasPedal1 > gasPedal2	rGP	releaseGasPedal	Soltar acelerador
gasPedal=1	pGP	puhGasPedal	Pisar al máximo
brakePedal1 < brakePedal2	pBP	pushBreakPedal	Pisar freno
brakePedal1 > brakePedal2	rBP	releaseBrakePedal	Soltar freno
Sw1 < Sw2	SW2R	Moving Steering Wheel to Right	Volante derecha
Sw1 > Sw2	SW2L	Moving Steering Wheel to Left	Volante izquierda
clutchPedal1 < cluthPedal2	pCP	pushClutchPedal	Embragando
clutchPedal1 > cluthPedal2	rCP	releaseClutchPedal	Desembragando
Gear1 < Gear2	G+	changingGearUp	Marcha superior
Gear1 > Gear2	G-	changingGearDown	Marcha inferior
Gear=7	GR	changingGearRear	Marcha atrás

Tabla 88: Reglas eventos atómicos

ANEXO D: RESUMEN EN INGLÉS

En este anexo se realiza la traducción en inglés de diferentes secciones del documento, destacando el capítulo de “Introducción”, “Conclusiones” y “Resultados”.

D.1.Introduction

Computer Vision is one of the areas of Artificial Intelligence that has grown more in recent years. It´s based on the study of how to process, analyze and interpret images with the goal to get certain information that helps to solve real world problems [1] .

One of the challenges of computer vision is the recognition and classification of human actions or activities from the frames that make up a video stream, using pattern recognition techniques. The activity human recognition by vision is composed by three distinct tasks: video pre-processing, representation of visual information and machine learning, combined with statistical techniques for detecting behavior patterns [2].

One of the main application areas of computer vision systems is the advanced driver assistance security, in which the goal is to recognize the activity or action that is being performed by the driver at that moment, using cameras and sensors. A system that is able to recognize which activity is performing the driver is very useful because it can prevent a lot of accidents caused by distractions. Other application areas are in medicine, security or industry.

D.1.1. Motivation

Traffic accidents are a real drama for the current society. In the last ten years it has reduced the number of accidents by 65% (see Figure 1). However, the number of accidents and deaths is still quite high. During 2014, in Spain, there were a total of 1131 deaths in 981 highway accidents [3].

Figure 1: Traffic accidents evolution in Spain [4]

Numerous studies have shown that approximately 78% of accidents are caused by driver's distraction, which can be physical (distractions with other passengers or other cars) or cognitive (stress distractions or perform a task too boring) .

A system that is able to recognize if a driver is distracted (e.g. watching radio) could avoid a lot of accidents and save many lost lives on the road.

D.1.2. Description of the problem

The research group CAOS (Control, Learning and Systems Optimization) of the Carlos III University of Madrid is developing a system to detect behavior patterns of a driver. The implementation of sensors in a real vehicle is a very expensive task. Besides, to experiment in a real car would be very dangerous. Thence, the CAOS group has acquired a car simulator called STISIM [5] (see Figure 2), which simulates a normal driving of a person in real time. This simulator includes a lot of simulated sensors to get data of it.

Figure 2: Simulator environment

This simulator has a lot of sensors to record all the information generated during the process of driving. These data are, among others, the angle of rotation of the wheel at each instant of time, the level of tread clutch and brake, how far is the front and rear vehicle, etc.

Besides, the group also has a XBOX360 Kinect camera (see Figure 3), which allows the position detection of face and eyes of the driver using an algorithm developed by the group of Intelligent Systems Department of Automatic [6] .

Figure 3: Kinect camera

The problem is based on to study and to develop a theoretical and practical activity model that defines the different activities that can be performed while driving a vehicle, performing a first practical model that is able to detect behavior patterns of a driver in real time in the future, using different sensors and cameras.

D.1.3. Scope of the project

The main goal of this project is to develop a theoretical and practical activity model that provides the basis of a practical model to recognize driver's real time behavior patterns.

The specific goals of this project are:

• To develop a theoretical and ideal activity model of driving.

• To get different data from cameras and simulated sensors that are similar to the data defined in theoretical model.

• To implement a program to process, transform and reason with the data.

• To develop a first approximation of a practical model that recognizes any proposed activity.

D.1.4. Memory organization

Chapter 1 includes the introduction of this project, emphasizing the motivation to do it, in the description of the problem and the general and specific goals of this project. At the end of this chapter, the organization of the memory is defined.

In the Chapter 2, the state of the art is developed, that is, the current state of computer vision and activity recognition by sensors is developed and exposed. Next, the ADAS (Advanced Driver Assistance Systems) are exposed. Finally, the main applications of the activity recognition are explained.

In the Chapter 3 a theoretical ideal activity model is developed in the driving field. Firstly, the data viability is studied, that is, which data can be extracted from the camera and the sensors. At the end of this chapter, a hierarchical theoretical activity model is proposed. Each activity is theoretically developed.

In the Chapter 4, the analysis and design of the system is developed. The system is able to process, transform and reason with data generated by the camera and the sensors. Firstly, the rules and restrictions of the system are developed. Next, the operational environment of this project is explained. Afterwards, the requirements and uses cases are specified. Finally, the architecture (design) of the system to implement is defined.

In Chapter 5, the implementation of the system is developed, describing with great details the processing, transformation and reasoning of the data generated. Also, a UML

diagram is included, showing the relationships between the different classes, methods and variables that make up the system code. Finally, the experimentation is detailed.

In Chapter 6, the planning and budgeting of this project are defined, making a Gantt chart for planning and a study of materials and human costs for the budget of this project.

In Chapter 7, the conclusions of this project are defined. Besides, the future works of this project are explained.

Finally, the document includes the glossary, where definitions of keywords and references can be found. Also, bibliographical sources and annexes with English summary are presented. Finally the user manual is exposed.

D.2. Conclusions and future works

In this section, technical and personal conclusions are explained , as well as the future lines of this project.

D.2.1. Technical conclusions

Technical conclusions are the following:

- Main goals and specific goals have been achieved:
  - An activity model ideal has been performed in order to apply it to practice later. This model consists of the theoretical definition of the different events, activities, tasks and maneuvers that the driver can perform in a vehicle.
  - A system that is able to get data from cameras and sensors has been implemented. Besides, this system is able to process, transform and reason with such data in order to recognize atomic events.
  - A first approximation of a practice task model has been developed. This model is compared with the theoretical model and good results have been achieved.

- After the experimentation performed, the transformation of the theoretical model to a practical model, which is able to recognize activities, has become a complex task. It depends of several factors, like concurrent events and time, which cannot be defined in the theoretical model.

- The extraction, process and transformation of data are some of the most important aspects of this project. It's important to study what variables are necessary for each environment, as well as the transformations that are necessary to apply to the data for a better study of the data. A task log file with no representative variables is a file without useful information, so that its's fundamental choosing the correct variables.

D.2.2. Future lines

In this section, the future lines of this project are studied. Next, the most important future lines are explained:

- *Improve the application's eyes and face position detection:* in this project, the position detection of face and eyes is not performed very well. Sometimes, the application is not able to detect the positions and returns "no-detected". For a good performance of the application, it's necessary that it detects the first images correctly. If the first images are not correctly detected, the rest of the execution will fail.

- *Improve the theoretical activity model:* the theoretical activity model is very complete, but it could be improved. For example, some atomic events or different tasks could be defined. Besides, these events can make up some tasks.

- *Observe more states on the log:* there are some states in the theoretical model, like "huecoAparcar=true", which are not directly observable in the simulator. These states can be observed on the simulator with some calculations.

- *Generate more log files:* instead of generating only one log file for each defined task, several log files could be generated in order to perform a more accurate task model.

- *Generate a trie by task*: for each defined task, generate a different trie, considering the time. This means keeping those atomic events that repeat over time.

- *Use different tries:* it is interesting to discriminate the information to be included in the trie. In this project, the information about head and eyes position is not included in the trie. In a future line, it would be interesting to generate different tries for each task. For example, one trie could include driver's data. Another trie could include car data and another could include driver's head and eyes position.

- *Perform a maneuver model:* once generated the improved task model, a maneuver model could be created, which could recognize what maneuver is performed, based on a set of tasks.

- *Delete the manual preprocess*: for that, it would be necessary to improve the system code, generating a method that is able to delete all the useless information from the log files.

D.2.3. Personal conclusions

Regarding the personal conclusions, I am very satisfied with the implementation of this project, because all the proposed goals have been achieved. I had never faced a project of this size, so I learned how difficult it is to plan and realize a project of great complexity.

My knowledge about the activity recognition domain was null before performing the project. The beginning of this project was quite complex because I had to read a lot of documents to understand the basic aspects of that domain. Once I understood the main issues about that domain, the development of the theoretical activity model was the hardest task.

In addition to acquiring knowledge in the area of activity recognition, I learned indirectly other concepts, such as C++ programming and budget developing.

Finally, I value very positively the interest in me of my tutor. He solved me every problems that appeared and he lead me in the project.

D.3. Results of the experimentation

In this section, the results of the experimentation of this project are presented. As detailed in previous sections, once generated and processed a log file of an activity, this file is labeled, instance by instance, with the atomic events that are detected. Thus, there is a data set composed of the atomic events that represents a task. The “Figura 60” shows the experimentation performed to label atomic events on the task “Detenerse”.

D.3.1. Practice approach proposed

The proposed approach to perform a first approximation of a practical model is based on [21].

Starting from a data file that is associated with a task, it concludes that the sequence of atomic events recognized in the file represents the task itself. A sequence of atomic events {A1, A2, A3 .., AN} is a set of atomic events sorted in time.

The goal is to get the most relevant sequences of events, as they will accurately describe the sequence. In the future, a task could be identified based on its most relevant sequences of events.

The proposed approach is based on the assumption that the most relevant sequences of events are those which repeat more often. This is already done in areas such as text mining, where the most relevant words in a text are detected based on their frequency within it.

To get the sequence of the most relevant events from a task, a tree-shaped structure is used, called trie, which identifies the most repeated atomic events and their dependence before and after events in time. A trie is a tree-shaped structure in which data is orderly inserted into nodes. Each node represents a subsequence of events. All subsequences has a starting node called root (see Figure 4). This structure is organized by level (depth), which means that to reach a node of level 2 is necessary to pass through a node of level 1.

For example, given the sequence {ls, date, ls, date, cat} where depth=3 is selected, the set of generated subsequences with that size would be: {ls, date, ls}, {date, ls, date}, {ls, date, cat}.

Next, the subsequences of each subsequence are inserted in the trie. For example, for the first subsequence, the following subsequences are inserted: {ls,date,ls}, {date,ls} y {ls}.

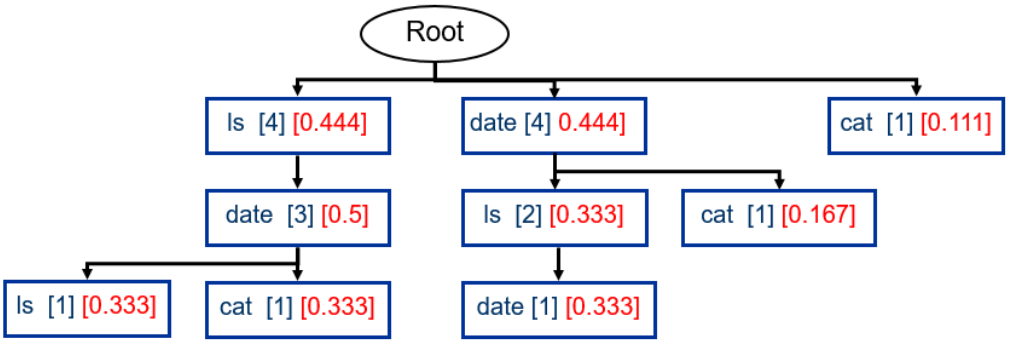

Figure 4: Trie [21]

Whenever a subsequence is inserted, insertion frequency (in blue brackets) is increased. Furthermore, the red bracket indicates the relevance of that subsequence in the final model. In the example, the most relevant subsequence is {ls, date}, with a importance (support) value of 0.5.

Once the sequences relevance is obtained, the goal is to generate a descriptive trie, i.e. a trie that identifies those sequences of atomic events that describe a task.

Next, the process performed to generate a first approximation of a model about the tasks "Stop" and "Start and Exit" is described.

D.3.2. Task model: stop

For this first approach, it is decided to study the task "Stop". Is remembered that the task is based on the following atomic events (see Table 10):

- *releaseGasPedal (rGP)*
- *pushBreakPedal (pBP)*
- *clutched (pCP)*
- *changingGearDown (G-)*

In addition, as described in the theoretical ideal activity model, the vehicle must pass through the states of "decreasingRPM" and "carBraking".

It is important to note that in this task are not involved atomic events related to the position of driver's face and eyes, so it is decided not to include these events in the model, as the complexity of the experiment would increase, generating "noise "in the data.

To reduce the gap between the theoretical model and practical model, it is decided to also remove those sequences that are repeated over time. For example, if the atomic event "pushGasPedal" is recognized in 50 instances, then it means that the activity is being performed continuously over time. However, the variable "time" is not considered within the defined theoretical model, so it is interesting to contemplate only the atomic events that indicate a change in the driver's behavior, in order to be able to recognize the different atomic events that make up a task. Including a large number of repeated sequences (atomic events repeated over time) means that the most relevant sequences of a task are equal, not being able to recognize the rest of atomic events that compose the task.

To apply the trie, an application developed by a member of the research group CAOS (Control, Learning and Systems Optimization) of the Carlos III University is used (see Figure 5).

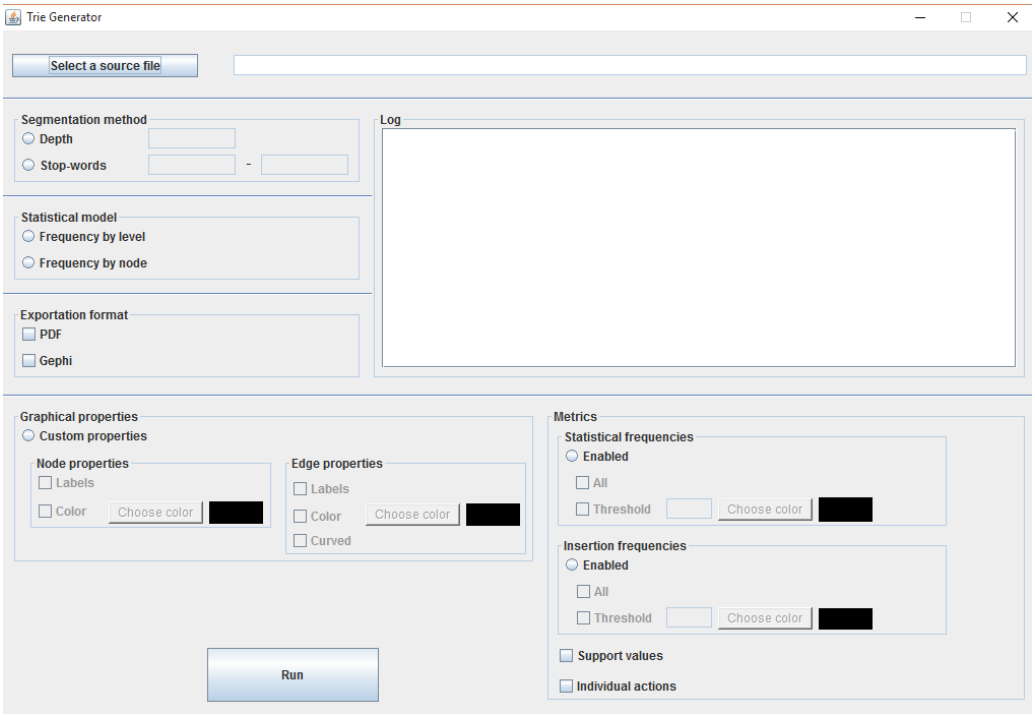

Figure 5: Trie generator

In the field “Source File”, the file of task “Detenerse” generated by the system is established. It is necessary replace the character “-“ by another character (in this case, by “:”), because the application uses the hyphen (“-“) to separate sequences and it not accepts input files with that character.

After trying various configurations, as a depth value (maximum length of subsequences), the value 5 is set for this first approach.

Because is desired to extract the value relevance of each subsequence (support), the option of "Support Values" is marked. After running the program, all subsequences are generated with its associated support value. The subsequences with greater support value are:

Support	Subsecuencia
0.192	Root-cB_:RPM_pCP
0.190	Root-cB_:RPM
0.123	Root-cB_:RPM_pBP_pCP
0.115	Root-cB_:RPM_rBP_pCP
0.100	Root-cB_:RPM_pBP

Table 1: Support subsequences

The most relevant sequences have as atomic events "carBraking (CB)", "RPM" (decreasingRevolutions), "pushBreakPedal (PBP)" and "pushClutchPedal (PCP)", and these events are concurrent in all cases.

When stopping a car, the car is slowing down, so this first model approximates quite rightly to the ideal. To reduce gear, the driver must depress the clutch constantly (event "pCP").

It is normal that not appear as relevant sequence, the event of gearing down ("changinGearDown"), as this event only occurs 3 times in all original log (The event is only recognized when changing from a higher gear to a lower gear).

Although the most relevant sequences are in depth one, such sequences are formed by concurrent events. That means, each sequence includes a set of events occurring at the same time instant.

Thus, it is concluded that the sequences with the highest support reflected in the table are the sequences that dominate the task, that is, the practical model future should be able to identify the task based on those atomic events that make up the sequences.

It is decided to choose and maintain the depth five (depth=5) because other configurations have produced worse results.